

ARIZONA DEPARTMENT OF TRANSPORTATION

REPORT NUMBER: FHWA/AZ 86/211

IMPLEMENTATION OF ON-RAMP TRAFFIC CONTROL ON THE BLACK CANYON FREEWAY

Prepared by:

Subramaniam D. Rajan
Jack Blackburn
James Lien
Venkat Subbarao
College of Engineering and Applied Sciences
Arizona State University
Tempe, Arizona 85287

APRIL 1986

Prepared for:

Arizona Department of Transportation
206 South 17th Avenue
Phoenix, Arizona 85007
in cooperation with
U.S. Department of Transportation
Federal Highway Administration

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Arizona Department of Transportation or the Federal Highways Administration. This report does not constitute a standard, specification, or regulation. Trade or manufacturer's names which may appear herein are cited only because they are considered assential to the objectives of the report. The U. S. Government and the State of Arizona do not endorse products or manufacturers.

TECHNICAL REPORT DOCUMENTATION PAGE

1. REPORT NO. FHWA/AZ-84/211		2. GOVERNMENT ACCESSION NO.		3. RECIPIENT'S CATALOG NO.	
4. TITLE AND SUBTITLE On-Ramp Traffic Control On the Black Canyon Freeway				5. REPORT DATE March 1986	
				6. PERFORMING ORGANIZATION CODE	
7. AUTHOR(S) S.D. Rajan, J.B. Blackburn, J. Lien and V. Subbarao				8. PERFORMING ORGANIZATION REPORT NO.	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Advanced Research in Transportation College of Engineering and Applied Sciences Arizona State University Tempe, Arizona 85287				10. WORK UNIT NO.	
				11. CONTRACT OR GRANT NO. HPR 1-27(211)	
12. SPONSORING AGENCY NAME AND ADDRESS Arizona Department of Transportation 206 S. 17th Avenue Phoenix, AZ 85007				13. TYPE OF REPORT & PERIOD COVERED Final Report February 1985 - Nov. 1985	
				14. SPONSORING AGENCY CODE	
15. SUPPLEMENTARY NOTES Prepared in cooperation with the U.S. Department of Transportation, FHWA. The opinions and conclusions are those of the authors and not necessarily of ADOT or FHWA.					
16. ABSTRACT This research implemented the ramp management strategy (RMS) developed in a previous study, on the Black Canyon Freeway. The software system developed by Safe-Trans was modified to include the RMS algorithm for controlling the entry of ramp vehicles according to speed-volume-occupancy relationships on the freeway. Decisions on metering rates were made at each ramp location on a minute-by-minute basis and may depend on the volume, percentage occupancy and average speed on the outside freeway lane. Limited field tests were conducted to verify the capabilities of the system. This report contains the details on the RMS, Nova III computer system, ramp controllers and program listings.					
17. KEY WORDS Ramp Metering, Ramp Management, Traffic Control				18. DISTRIBUTION STATEMENT No restrictions	
19. SECURITY CLASSIF. (of this report) Unclassified		20. SECURITY CLASSIF. (of this page) Unclassified		21. NO. OF PAGES 74	
				22. PRICE	

Acknowledgements

The authors would like to thank the personnel at the Arizona Department of Transportation for their cooperation in making the project a success. Special mention should be made of Bud Smith who helped with the operations of the Safe-Trans System, and David Olivarez, Ron Midkiff and David Johnson for their invaluable assistance. Thanks are also due to the personnel in the Traffic Control Devices Shop for their assistance during the field tests.

Finally, the help and cooperation of Frank McCullagh and Rudolf Kolaja of ATRC is gratefully acknowledged.

Table of Contents

	<u>Page</u>
Chapter 1	OBJECTIVES OF PROPOSED RESEARCH 1
Chapter 2	OVERVIEW OF ORIGINAL SOFTWARE 5
	Description of the database files 6
	Description of main modules 12
	Utility programs 15
	Capabilities of NOVA III 18
Chapter 3	RAMP MANAGEMENT STRATEGY 31
	Positioning of controllers & detectors 31
	Description of controllers 31
	The ramp management strategy 40
Chapter 4	MODIFIED SOFTWARE 43
	Common Blocks 43
	Modifications to Different Modules 45
	RMPT
	COMMT
	DATAP
	New routines 48
Chapter 5	OPERATING PROCEDURES 51
	Bootting the system 51
	System changes and enhancements 53
	Directories and files 53
Chapter 6	CONCLUDING REMARKS 54
	Impact on freeway traffic 54
	Future enhancements 55
	REFERENCES 57
Appendix	PROGRAM LISTINGS 58

List of Figures

<u>Figure</u>	<u>Title</u>	<u>Page</u>
2.1	General format of file ISFILE	9
2.2	Flow diagram of Ramp Controller System	16
3.1	Detector placement	32
3.2	Schematic diagram of ramp controller	34
3.3	Ramp Metering Strategy flow diagram	41
4.1	Revised ramp metering strategy	47

List of Tables

<u>Table</u>	<u>Title</u>	<u>Page</u>
3.1	Ramp Metering Timing Card	35

CHAPTER 1

OBJECTIVES OF PROPOSED RESEARCH

As a part of the research project "Design of On-Ramp Traffic Control on the Black Canyon Freeway," Arizona State University developed a new algorithm for controlling the entry of ramp vehicles according to speed-volume-occupancy relationships on the freeway as well as providing for a ramp closure in the event of a non-recurring incident [1].

Decisions on metering rates are made at each ramp location on a minute-by-minute basis and they depend on the volume, percentage occupancy, and average speed on the outside freeway lane.

The effects of the ramp metering strategy were not measured directly in this project. In the absence of direct information on the effect of different metering rates on freeway volume, an on-site test of the strategy and final adjustment of criteria for satisfactory freeway operation needs to be made.

Objectives

The objective of this research work is to implement this management strategy for controlling on-ramp traffic volumes which would minimize the chances of exceeding critical freeway lane volumes and the resulting congestion and stoppage of the freeway. Specifically, this research focuses on accomplishing the following:

1. Implementing the flow diagram for ramp management strategy (RMS) on the NOVA III system.
2. Modifying the existing software developed by Safe-Trans [2] so that appropriate information can be sent from the NOVA to the ramp controllers and back.

3. The programming style of the RMS includes external control of:
 - a. The upper and lower speeds and percentage of freeway occupancy
 - b. The length of time in minutes for volume-occupancy-speed values to remain in the monitor range before ramp closure.
 - c. The values of percentage occupancy which define the boundaries for metering rates.
 - d. The metering rates for fixed upper and lower values of percentage occupancy.
 - e. The minimum one-minute value of outside freeway lane volume for metering to begin.
 - f. The maximum one-minute value of outside freeway lane volume which, if exceeded, causes ramp closure.
 - g. The minimum one-minute value of outside freeway lane speed below which a ramp is closed.
 - h. The maximum one-minute value of outside freeway lane percentage occupancy which, if exceeded, causes ramp closure.

Background and Significance of Work

Stoppage of morning traffic flow on the southbound lanes of the Black Canyon Freeway is a frequent occurrence during the months of heaviest traffic flow. Such stoppages may be caused by an isolated incident involving a single vehicle. Or, they may be caused by traffic volumes which approach or exceed the carrying capacity of the freeway.

When freeway volumes approach capacity, any number of events may cause slowing or stoppage of traffic. Lane changes in the vicinity of the interchanges, slowing to avoid exiting or entering vehicles, and differing performance characteristics of both drivers and vehicles, all contribute to freeway congestion. This can lead to a significant reduction in average

freeway speed and may, in turn, cause traffic flow to 'grind to a halt' for periods ranging from seconds to several minutes and may occur repeatedly until the excess volume is finally dissipated and flow returns to normal.

Satisfactory speed-volume relationships can be maintained on the freeway if entering ramp volumes are controlled at each interchange to prevent freeway lane volumes from exceeding a critical level beyond which congestion and stoppage are likely to occur. The volume of ramp traffic entering the freeway can be controlled by metering the rate at which ramp vehicles are allowed to enter the freeway.

The California Department of Transportation conducted a study on freeway incident detection. While the California study reported the results of tests on many algorithms for incident detection it did not provide a strategy for controlling traffic volumes until traffic flow could be restored to normal. The management strategy for controlling on-ramp traffic volumes that would minimize the chances of exceeding critical freeway lane volumes and the resulting congestion and stoppage of traffic was carried out by Professor Blackburn [1]. The objective of the proposed study was to implement this strategy.

This report is divided into six chapters and an appendix. The original software developed by Safe-Trans is discussed first. The details on hardware and software are presented briefly. Next, the ramp management strategy (RMS) is discussed with a view towards the modified software. Changes to the existing software are presented in detail. The operating procedures follow next. The step-by-step procedure from booting the computer system to configuring the RMS is presented. The emphasis is on easy user-system interface and all new software is designed to be user-oriented. Finally, details on the system testing are presented. The report

concludes with a discussion on the impact on freeway traffic due to ramp metering and possible system enhancements.

CHAPTER 2

OVERVIEW OF ORIGINAL SOFTWARE

The original software developed by Safe-Trans Corporation was designed to provide information on traffic flow on the Black Canyon Freeway. A Data General Nova III minicomputer, connected to controllers and loop detectors via telephone (communication) lines, gathered information at fixed intervals (usually 5 mins) from the 19 ramps. The information on speed, volume and occupancy can be printed in a tabular form or displayed graphically on a Tektronix graphics terminal.

A historical database is continually updated and stored by the system with information for the preceding 31 days residing on the computer disk for easy preview. In addition, the system also provides a mechanism to either read or write data on the ramp controllers. Such data include the cycle lengths for red and green signals, the detector volume counts, start time and end time for metering operation. If and when a detector fails, the detector code and time of failure are recorded in a detector status database.

In this chapter, the overview of the original software is presented. First, the database files are explained in terms of their format and functions. These files are used by one or more modules for data processing. Second, the main modules themselves are explained. These modules continually run in a predefined sequence gathering information from the ramps and communicating with the ramps and peripheral devices when required to do so by utility programs. Some of the utility programs are explained next. They display data or communicate with the ramp controllers. Finally, the capabilities of the NOVA III computer are discussed. The text editor is explained in detail since some system parameter changes or the development

or modifications of modules require the usage of the editor. The procedural steps for the creation of FORTRAN files, their compilation, linking (or loading) and execution are also explained.

2.1 Database Files

The current software for the ramp metering computer system abstracts information from five important database files for processing information essential to the ramp management strategy for decision making. The files are named as below:

1. DATABASS - Database file for southbound ramps
2. DATABASN - Database file for northbound ramps
3. ISFILES - Internal storage file for southbound ramp
4. ISFILEN - Internal storage file for northbound ramp
5. HISn - History storage file (n: 0 to 31)

DATABASS and DATABASN are created by the user and provided to the system, whereas the other database files, namely ISFILES, ISFILEN and HISMK are opened and updated by the software at a pre-fixed time and in a designated format. A description of each database file is given below.

DATABASS & DATABASN

These two files are used by the Main Line Status Report (MLSR) program to plot volume, occupancy, and speed information as an analog bar chart on a Tektronix graphic terminal. The database file for north and southbound ramps depict a similar format. The general format of this database file is as follows:

NUMI	D1	CT	RS	MF	IBAUD
INN1	NL1	SC1	IN1		
INN2	NL2	SC2	IN2		
-	-	-	-		
-	-	-	-		
INNn	NLn	SCn	INn		

The above abbreviations may be expanded as:

NUMI = Number of ramps in the database file for which the report is to be prepared by MLSR.

D1 = Average vehicle length in feet

CT = Occupancy clock frequency in Hertz (cycles/sec)

MF = Constant for converting speed units from miles/hr to kilometers/hr.

IBAUD = Transmission rate to graphic terminal in characters/sec (i.e., Baud rate divided by 10)

INNn = nth ramp name

NLn = Number of lanes in the nth ramp

SCn = Normal signal cycle length in seconds for the nth ramp

INn = Ramp number of the nth ramp

ISFILES & ISFILEN

These two files are created by the Interval Storage Maker (ISMK) program every interval time specified when booting the system ("INTERVAL IN MINUTES = ? (1-180)"). Database files are named as IST files during the first interval time, after which new IST files are opened and the previous IST files are renamed ISFILES and ISFILEN. On the completion of the second interval time, again new IST files are opened and the previous IST files are

renamed ISFILES and ISFILEN. ISFILES and ISFILEN which were present previously are renamed ISFILES1 and ISFILES2.

The general format of ISFILE is as shown in Fig. 2.1.

The abbreviations shown in Fig. 2.1 are listed below:

First Line

IMO = Month of the year

IDY = Day of the month

IYR = Year In which the file

IM = Hour of the day was created

IMN = Minute of hour

ISEC = Second of the minute

IMI = Used as a test bit

IDI = Ramp number for which data is filed

Second Line

IF1 & IF2 = volume parameters collected by two frontage detectors

IR = volume parameter collected by the ramp detector

IC2 & IC1 = volume parameters collected by the two check-in detectors

IQ = volume parameter collected by the queue detector.

IS(I+2),
IS(I+1),
IS(I) = volume parameters collected by the three secondary detectors

IP(I+3),
IP(I+2),
IP(I+1),
IP(I) = volume parameters collected by the four primary detectors

Third Line

IOF2,
IOF1 = occupancy parameters collected by the two frontage detectors

IMO	IDY	IYR	IM	IMN	ISEC	ISI	IMI	IDI					
IF2	IF1	IR	IC2	IC1	IQ	IS(I+2)	IS(I+1)	IS(I)	IP(I+3)	IP(I+2)	IP(I+1)	IP(J)	
IOF2	IOF1	IOR	IOC2	IOC1	IOQ	ISP(I+2)	ISP(I+1)	ISP(I)	IOP(I+3)	IOP(I+2)	IOP(I+1)	IOP(I)	IPCTV

Fig. 2.1 General format of file ISFILE

- 3) Second of the minute
- 4) Last history time in minutes
- 5) Flag for metering or nonmetering
 - 1 --> metering
 - 0 --> nonmetering
- 6) Ramp number for which the data is filed

Second Line

- 1) First frontage detector volume parameter stored in history buffer
- 2) Second frontage detector volume parameter stored in history buffer
- 3) Ramp detector volume parameter stored in history buffer
- 4) First check-in detector volume parameter stored in history buffer
- 5) Second check-in detector volume parameter stored in history buffer
- 6) Queue detector volume parameter stored in history buffer
- 7) First secondary detector volume parameter stored in history buffer
- 8) Second secondary detector volume parameter stored in history buffer
- 9) Third secondary detector volume parameter stored in history buffer
- 10) First primary detector volume parameter stored in history buffer
- 11) Second primary detector volume parameter stored in history buffer
- 12) Third primary detector volume parameter stored in history buffer
- 13) Fourth primary detector volume parameter stored in history buffer

Third Line

The third line is exactly similar to the second line, except that instead of volume parameters, occupancy parameters are stored for the respective detectors in the history buffer.

2.2 Description of Main Modules

1. RMPT

Ramp Communication initializer

This program is used to get the initial data from operator. It will ask five questions when the system just booted up. They are listed below.

1. "DAY OF THE WEEK IS ? (0-6 SUNDAY-SATURDAY)"
2. "INTERVAL IN MINUTES=? (1-180)"
3. "FIRST INTERVAL IN MINUTES=? (1-180)"
4. "HISTORICAL TIME IN MINUTES=? (5-180)"
5. "FIRST HISTORICAL TIME IN MINUTES=? (1-4320)"

The answer for the second question is used to determine the interval time of collecting data from the ramp controller and hence the interval data must be compatible with the ramp controller, i.e., the data in address FEF4 of ramp controller must have the same value as the answer times 60. Normally this is set to 5 minutes for nonmetering mode and to 1 minute for metering mode of operation.

Historical time is used for the interval time of historical file to be stored. Normally this is set to 15 minutes.

The first historical time should be set to the value. Initially the database files are read and data initialized. RMPT then will chain to COMMT when the minute changes. In Fortran V, it is represented by the statement "CALL FCHAN("COMMT.SV")." Before chaining to COMMT, common data is saved by SCOMM, SDATW, PASCW routines.

2. COMMT

Communication Handler

This is a communication generating program. It sets up messages for the various ramps, and reports communication failures to ERRORSTOR.

In the beginning, SCOMR and PASCOR are called to pass the values of the common block. Subroutine PRNT is called to check if the background program is asking for printing or graphing or ramp communication. If a print request exists then the background's request is executed by PRNT subroutine. For the next 20 cycles, COMMT checks the status flag to see if communication data is to be sent to any of the ramps. If message is needed for one particular ramp, then the message is formatted by subroutine RMT. SXMIT is the next subroutine to be called to perform two tasks. One is to receive the collected data from the detectors in order to calculate the volume, occupancy and velocity, the other is to send the message to the ramp controller to read or write data. Within the loops, COMMT keeps checking to see if time has changed. If it has, then COMMT chains to DATAP. Before chaining, SCOMW and PASCW are called again to save the common data into files.

Routines Ramp Response is used to pack up the collected data into IDAT in order to be used in DATAP, and Comm Error is used to store communication errors, if any.

3. DATAP

Data Processor

This program generates IST files which will be renamed to the ISFILEN/S the next time DATAP is executed. ISFILEN/S are used in the MLSR program for calculating volume, occupancy and velocity. DATAP also creates HSMK , which will be appended to HIS⁰ for every historical interval. The HIS⁰ file will be renamed to HIS¹ when the data is changed and the HISTORY program is chained. ERSTOR is called in this program to write the error message to the SYSLOG.

The first 60 statements write messages to the SYSLOG. The next DO loop update the data buffer and reset the flag for the next cycle. ISMK subroutine is for creating IST file, DATAP calls it twice to get the northbound and southbound, respectively. HISMK subroutine is for creating HSMK , which will append to HIS . CLKUP reads backup clock and compares it with system clock. If system clock is behind it is set equal to the backup. If system is ahead the date is advanced one day and the clock is then set equal to the backup. Finally, DATAP checks to see if the date has changed. If it is has then DATAP chains to HISTORY; else it chains to COMMT.

4. HISTORY

History Handler

This program basically does two tasks:

1. It moves and renames history files at midnight, and
2. It sets up clock synchronize messages to ramp controllers

PASCR, SCOMR and SDATR are called in the beginning to pass the common data, then HISTORY renames the 30 files to advance every data file by one day. The ramp clocks are then synchronized with the computer clock. Finally, HISTORY chains back to COMMT again.

The whole kernel for Ramp Control System is shown in Fig. 2.1.

2.3 Utility Programs

1. MLSR

MainLine Status Report

This program performs the tasks of graphing volume, occupancy and speed information in "near-to-real" time. Six files are used in this program:

1. DATABASN and DATABASS contain the information about average vehicle length, frequency of the occupancy clock (refer to description of the database file on page 6).

2. ISFILES, ISFILEN, ISFILES1 and ISFILEN1 contain information for the last sampling and the next to last sampling interval for northbound and southbound ramps.

MLSR reads the data from the above files and takes the average volume and occupancy from all lanes and converts it to vehicles per hour and percentage occupancy. Velocity is calculated from volume and occupancy. SPLR is a temporary buffer to contain the output from either graphics or file. The rest of the program deals with graphic functions and sets interground flag to queue output to the foreground. Note that MLSR calculates and draws the chart of volume, occupancy and

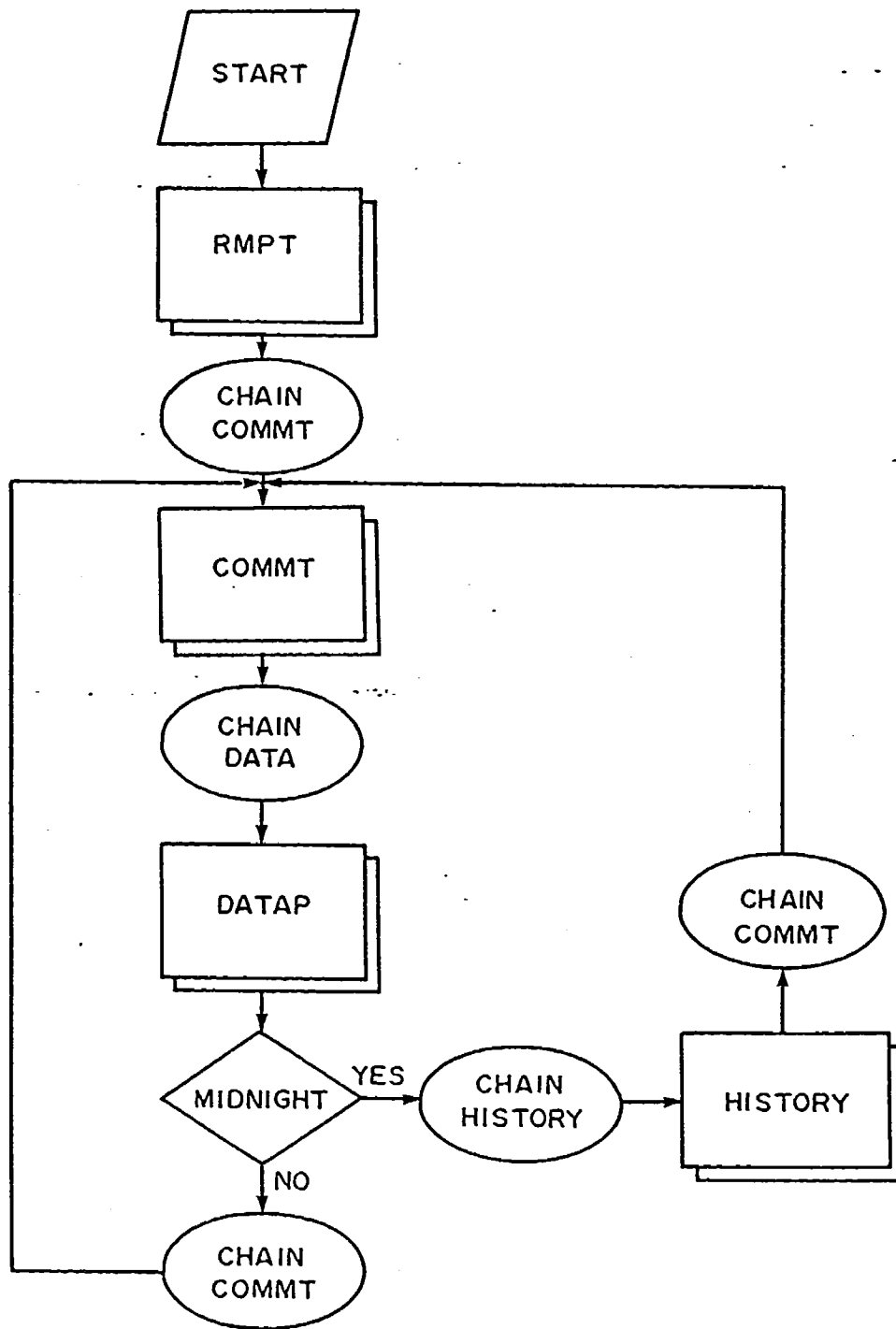


Fig. 2.2 Flow Diagram of Ramp Controller System

velocity according to the data of files ISFILES/N which are created by the foreground program DATAP.

2. RAMPTALK

Ramp Controller Communicator

This background program can read or write to ramp controllers. It runs in the background with RMPT, COMMT and DATAP running in the foreground. RAMPTALK packs the information into a file called "RDATA", then sets the interground flag to inform the foreground. The foreground program then opens the file to read those data into common block IRDATA. COMMT then uses SXMIT to send or receive the information to or from the ramp controller.

The operator is required to answer some questions in order to communicate with the controllers. The four answers needed are:

1. IRF (flag): if read then IRF=1, if write then IRF=2.
2. IRI (ramp number): 1010-1018, 1020-1029.
3. IRA (address): FE00-FEFF
4. IRD (data): refer to the controller instructions.

If the IRI is specified to be 1255, it means that the message is to be sent to all the ramp controllers.

3. RMPSR

Ramp Status Report

This program produces reports on either the CRT or the printer. These reports use information collected for the last interval and are contained in files ISFILEN and ISFILES. It prints out south- or northbound ramp status information and

occupancy information for the ramp and detector. The message which comes back from the detector will be -1 if the detector received no information. So, to decide if communication line is on, the program just checks to see if all the detectors send back -1 data.

2.4 Capabilities of NOVA III

NOVA III minicomputer used as the central processor in the ramp metering, is a 16 bit minicomputer. The total primary memory capacity of the NOVA III minicomputer is 128K words. The system further provides 10 megabytes of secondary memory on two disks, one fixed and another a removeable disk cartridge. Either of the two disks can store a maximum of 5 megabytes of information.

The real time disk operating system (RDOS) used by the NOVA III system provides the capability to program interactively. The RDOS also governs the allocation of cpu time to its two users, namely Foreground and Background. The ratio of the percentage of cpu allocation, to Foreground and Background is called the Foreground-Background ratio. This ratio may be changed, if the need arises. At present the RDOS allocates more cpu time to the Foreground, as various calculations and interaction between the communication routines and the system is achieved by this user. Other functions of the RDOS include disk stored data and program files manipulation.

In the present ramp metering system, the Safetran software is implemented by using a Datageneral CRT terminal as the background terminal and a Datageneral Dasher printer is used as the Foreground terminal.

Compiling a FORTRAN 5 Program Under RDOS

Each program must be compiled separately by issuing the following command:

FORTRAN filename

This command produces a relocatable binary file named filename.RB, with no listing and sends error messages to the console, if any.

The general form of the FORTRAN5 command line is:

FORTRAN [global switch] inputfilename filename [local switch] . . .

Some useful switches, both global and local have been listed below with their function.

Global switches

- | | |
|----|---|
| /B | Produce a brief listing. This includes the input source program, the list of all sub programs called the cross-reference, and the error list (but not the generated code). |
| /C | Check the syntax of the source program. If a listing file is specified, the source program and error list are sent to it. The error list is also sent to the error file, if one exists. |
| /D | Debug. It allows the error trace back routine to output line number. |
| /L | Produce a listing. If a listing file is explicitly specified using the local /L switch, the global /L switch has no effect. Global /L produces a listing filename, input filename.LS. |
| /N | Do not produce an object file. |

Local switches

- | | |
|----|--|
| /B | Direct the relocatable binary output to the file named. No listing is received of the generated code when this switch is used. |
| /E | Direct any error messages to the file named. |

/L Direct the listing to the specified file. The listing contains the input source program, a storage map, a list of all subprograms called, a listing of the generated code, a cross reference and a list of errors.

Loading FORTRAN5 Programs

After all programs necessary for a particular task have been compiled, and ascertained they are in the same directory, they can be loaded using the following command.

```
RLDR filename1, filename2, . . . @ FLIB @
```

In general, a program is loaded in the following sequence:

- 1) Main FORTRAN5 program
- 2) User subroutines

Subroutines must be entered in the order in which they are called from the beginning of the main program.

- 3) Support libraries

In this case we use only @FLIB@

The load command RLDR loads the main program and other routines and creates a save file, namely, main program.SV. If a program or a subroutine which is being loaded is not in the current directory, or is not compiled, an error message will be reported by the system.

Executing a Program

A program may be executed after it has been properly compiled and loaded by the system. This can be done by simply typing the filename and pressing the carriage return key. For example:

MAIN PROGRAM

Text Editor

The ability to key a program successfully into a computer or to make suitable correction in a pre-existing program is achieved by gaining dexterity over the text editor. The text editor commands can be explained broadly in two parts, namely: 1. To create a new file or open a pre-existing file for editing, and 2. Line editing of files. The present system uses NSPEED text editor for editing files.

1. Creating a new file

A new file can be opened by typing the following command and pressing carriage return.

```
NSPEED filename
```

where "filename" is the name of the file to be opened. The system responds to this command by typing out the message

```
CREATING NEW FILE  
!
```

The prompt symbol in the NSPEED mode is an exclamation mark, '!'.

Opening a pre-existing file

If a file already existing in the current directory is to be edited, the same command as above may be used. The system then responds to this command by opening the file named by "filename", and prompts.

Note: After a new file is created or an old file is opened using the above command, the system, before prompting, outputs the following line.

```
WARNING: EXPANDED ERROR TEXT NOT AVAILABLE
```

This may be disregarded as inconsequential, as all text editing commands in this manual lie within the four walls of the warning.

2. Line editing commands

Before the line editing commands are introduced it would be beneficial to understand the editing cursor, termed in this manual as the character pointer (CP).

The character pointer is an editing cursor, which may be viewed as a '^' on the console. Character pointer is always positioned between two characters, and never at a character. When a text is to be modified, the CP must be positioned exactly before the beginning of that particular line or word or character. For example,

- a) An insert command to insert letter 'T' when the CP is as shown below

S(^)OP

will insert 'T' as

STOP

- b) A search command for letter A when CP is in the beginning of the word

(^) DATA

will place the CP before

D(^)ATA

If again a search command is executed for letter A, the cp is placed in front of the second A in the word DATA.

DAT(^)A

So it must be remembered that any change is effective to only those strings or characters which follow the cp, and not those which precede it in a file.

For example, when cp is as shown below

S0(^)TOP

a change command intentional of removing the extra '0' from the word STOP will lead to the removal of the letter '0', following the cp, thus resulting in S0TP.

Escape character

The escape character serves two important functions.

1. A single ESCape, which is viewed on the screen as \$, is used to separate a single editing command or a command string from the next command character. For example,

J \$ T \$. . .

Editing
commands

Escape character used as
command separator

2. Two consecutive ESCapes, which is viewed on the screen as \$\$, is used as a carriage return in the NSPEED editing mode. It must be remembered that in the editing mode, the conventional carriage return is never used to execute the command. For example,

2L \$ 10T \$\$

Editing
commands

NSPEED
carriage
return

Command
separator

Hence the double ESCape tells NSPEED to execute the command string.
The ESCape character is keyed by pressing the ESC key on the keyboard.

Erasing characters while typing

The DEL key is used to erase any incorrect character detected while typing. Each time this key is pressed, NSPEED deletes the last character entered and echoes that character on the terminal. For example

```
! SUBROUTINE ENIT UTINE
```

```
1      2      3
```

Three steps may be understood as follows:

1. An error is detected after the word 'SUBROUTINE' is typed.
2. DEL key is pressed four times, which echoes the characters erased.
3. The correct string is entered.

If the DEL key is pressed more than the number of characters in a line, it echoes all the characters and gives a new prompt. For example,

```
! END DNE
```

```
!
```

Command structure

The general format of NSPEED command is as below:

[num-arg]...COMMAND[string arg]...\$\$

where:

num-arg is a decimal number

COMMAND is any NSPEED command

string-arg is a string of ASCII characters

Character pointer commands

The character pointer (CP) commands are necessary to move the editing cursor to that part of the program where correction is necessary. Using this command the CP can be moved forward or backward, by characters or by lines. The CP commands can be documented as follows.

COMMAND	DESCRIPTION
J	Move the CP to the beginning of the buffer's first line.
nJ {n is a numeric value}	Move the CP to the beginning of line n in the current buffer. n is always relative to line 1, the first line of current buffer.
nL	Move the CP n lines forward relative to its present position.
-nL	Move the CP n lines backward relative to its present position
L	Move the CP to the beginning of the current line
nM	Move the CP n characters to the right, relative to the current position.
-nM	Move the CP n characters to the left, relative to the current position.

Insert command I

Insert command is used to insert text in a newly opened file or to insert strings in a pre-existing file.

1. Newly opened file

After the prompt appears, an 'I' is typed and the desired text is keyed into the computer. For example,

```
      I IFORTRAN.....  
      _____  
      _____ $$  
      I
```

2. Pre-existing file

In a pre-existing file an insert command is usually used to insert text between two lines. This can be done by first moving the cp to the end of the line preceding the new line to be added. After this step, an I is typed and RETURN carriage return pressed. The desired text may be now entered. This can be shown by an example as below.

STEP 1:	FORTRAN SUBROUTINE (^) CALL LINK	A line is to be inserted between them
STEP 2:	II	I and carriage return
STEP 3:	INIT = EXEC \$\$	Text is entered
STEP 4:	FORTRAN SUBROUTINE INIT = EXEC CALL LINK	RESULT

Points to remember

1. Any number of lines may be inserted by typing the line and pressing RETURN key when desired.
2. When a FORTRAN code is to be inserted, it is important to realize that the FORTRAN commands start from the seventh column on the screen.

Therefore when an insert I is typed for the first line, the FORTRAN command must start from the eighth column, as I occupies the first column. For example,

123456789.....	
I I100 FORTRAN	First line
23 D010I=	second line

If confusion arises while counting the columns for entering the FORTRAN command in the first line, then sufficient space may be given and command started from the 9th or 10th column (or onwards), as FORTRAN accepts command starting from any column after the seventh column.

Search command S:

The search command searches for a particular string and places the CP after that string, if it is found. If more than one string of the same type exist, then it stops after the first string which follows the current position of the CP. Strings which precede the current position of the CP cannot be detected and will result in an error. It would be advisable to jump the CP to the beginning of the text before issuing a search command, if the approximate location of the string is not known. For example,

	J \$ S T A B L E \$\$	
CP Jump	string	NSPEED
command		carriage
	Search	return
	command	
	command	
	separator	

If a string is not found, the terminal displays the error message

ERROR: UNSUCCESSFUL SEARCH

The general format of search command is

Sxtring\$

Type-out command T

The function of the T command is to type out the current line of text, along with the symbol (^) which includes the current cp position. The T commands may be documented as below.

COMMAND	DESCRIPTION
T	Type out the current line where cp resides with (^)
nT where n is numeric	Type out n lines following the current position of cp
-nT	Type out n lines preceding the current position of cp.
#T	Type out the entire text in the buffer.

It must be remembered that the T command does not change the position of the cp.

Change command C

The C command is a combination search and modify command. General format of this command is

change command	C String1 \$ String2 \$	command separator
	string to be changed	new string

The change command searches the string1 from the current position of the cp and replaces when the string is found by string2. It must be

remembered that it will replace the very first string it finds, if there is more than one string of the same type. Therefore it is imperative to position the cp suitably before issuing the change command. An example of change command can be seen below.

```
! J $ FORTAN $ FORTRAN $ T $$
```

NSpeed positions the cp at the beginning of the buffer, then searches for FORTAN. NSpeed finds the string, inserts FORTRAN in its place, and types out the edited line.

Kill command K

Kill command is used to kill one or more lines following the current position of cp. The general format of the command is as shown below.

number of lines	n K \$	command separator
	kill command	

How to save a text after editing

If a file is to be saved after inserting text then Home command may be used. The general format of this command is:

```
! UEH $$  
R
```

This command saves the content of the buffer and exits from NSPEED at the end of an editing session, and returns to the CLI. If any files are open they are closed before returning to the CLI. If files were opened and no correction made, then it would suffice if the following command is issued.

I H \$\$
R

The R is RDOS CLI prompt. Once it appears, any CLI command can be issued.

CHAPTER 3

RAMP MANAGEMENT STRATEGY

The overall intent of the Ramp Management Strategy (RMS) is to achieve a smooth flow of traffic on the mainline freeway and ease the merging of traffic entering the freeway via the ramp, with the mainline traffic. The status of the traffic at any ramp is monitored by loop detectors placed at several locations on the freeway and the ramp. The ramp controller obtains the information from the loop detectors and stores the data. Upon request from the computer, it sends this information back for data processing.

In this chapter the locations of the detectors and controller are shown. The description of the controller follows next. The addresses and functions are explained via the Ramp Metering Timing Card. Using the information obtained from the controller, the formulae for the calculation of volume, occupancy and speed are outlined. Finally, the actual RMS and its simplified version are discussed.

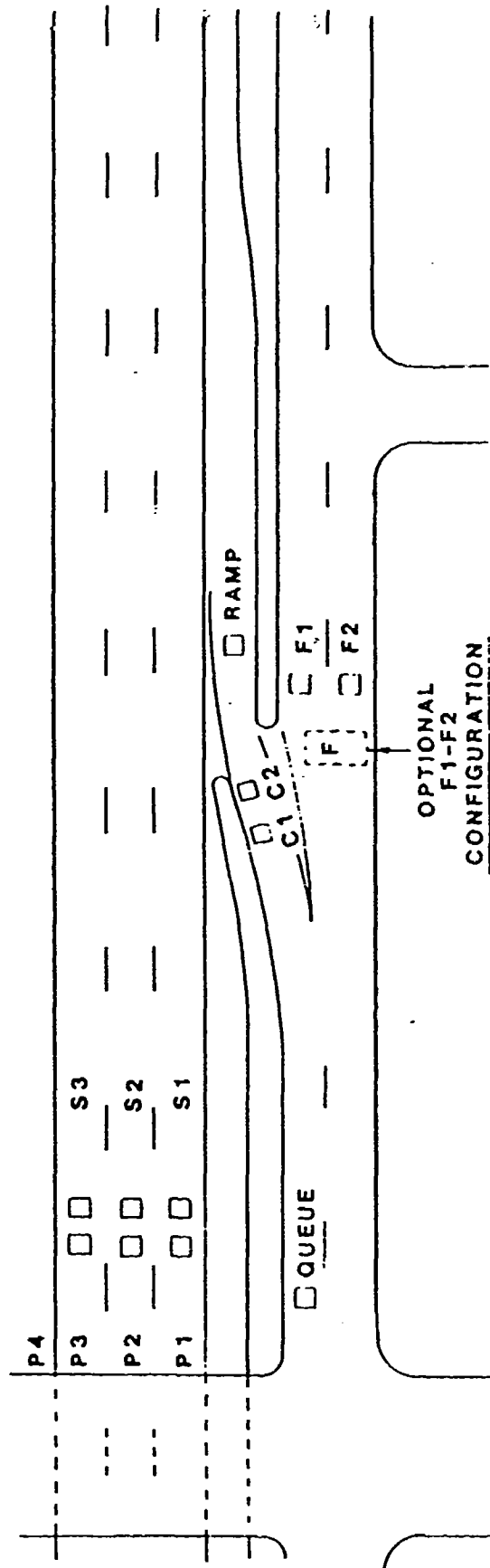
3.1 Positioning of Controllers and Detectors

The typical detector locations and identification labels are shown in Fig. 3.1.

The 13 mainline detectors are used to accumulate volumes and occupancies for each lane. In addition to volumes and occupancies, the detectors on the ramp provide the capability to accumulate data on the number of violations.

3.2 Description of Controller

In order to interface effectively with the ramp controller it is imperative to understand the functions it provides and the steps required to manipulate the variables which govern this function.



Detector Placement

Fig. 3.1 Detector Placement

The heart of the CR 1610 is a 16 bit microprocessor that performs the logical and timing functions of the controller. The parameters of the functions provided are stored in appropriate memory locations which can be accessed and manipulated. Functions necessary for the ramp management strategy and their respective addresses are described below.

When the toggle switch used in the front panel of the controller is turned on to DISPLAY & DET FAIL, it causes the seven segment LED to display the current value in memory selected by the CYCLE and INTERVAL thumbwheel switches. The details are shown in Fig. 3.2.

Figure 3.2 shows that the value at the memory location, cycle 6-Interval 2 is 0700. Each thumbwheel has 16 demarcations from 0 to 15, and hence providing for a total 16 x 16 addressable memory locations. All the values in the memory can be referenced at one time by tabulating them in a table, called Ramp Metering Timing Card, as shown in Table 3.1.

The general format of an address in this table is

FE CYCLE, INTERVAL

In the above example the address value in the hexadecimal system is FE62 (Fig. 3.2).

Important addresses

- FE00 --> FIRST YELLOW is used when the controller is starting to meter. It can be set to zero and the CR1610 will ignore this parameter.
Range: 0 to 799.9 entered in tenth of seconds.
- FE02 --> TIME OUT is the time the metering sign remains ON after the last red indication.
Range: 0 to 799.9 entered in tenth of seconds.
- FE03 --> LEAD IN is the time the metering sign is ON prior to displaying first yellow
Range: 0 to 799.9 entered in tenth of seconds.
- FE30 --> This is the total cycle time for a RED-GREEN cycle. It is used in conjunction with FE32 which provides the GREEN time in a

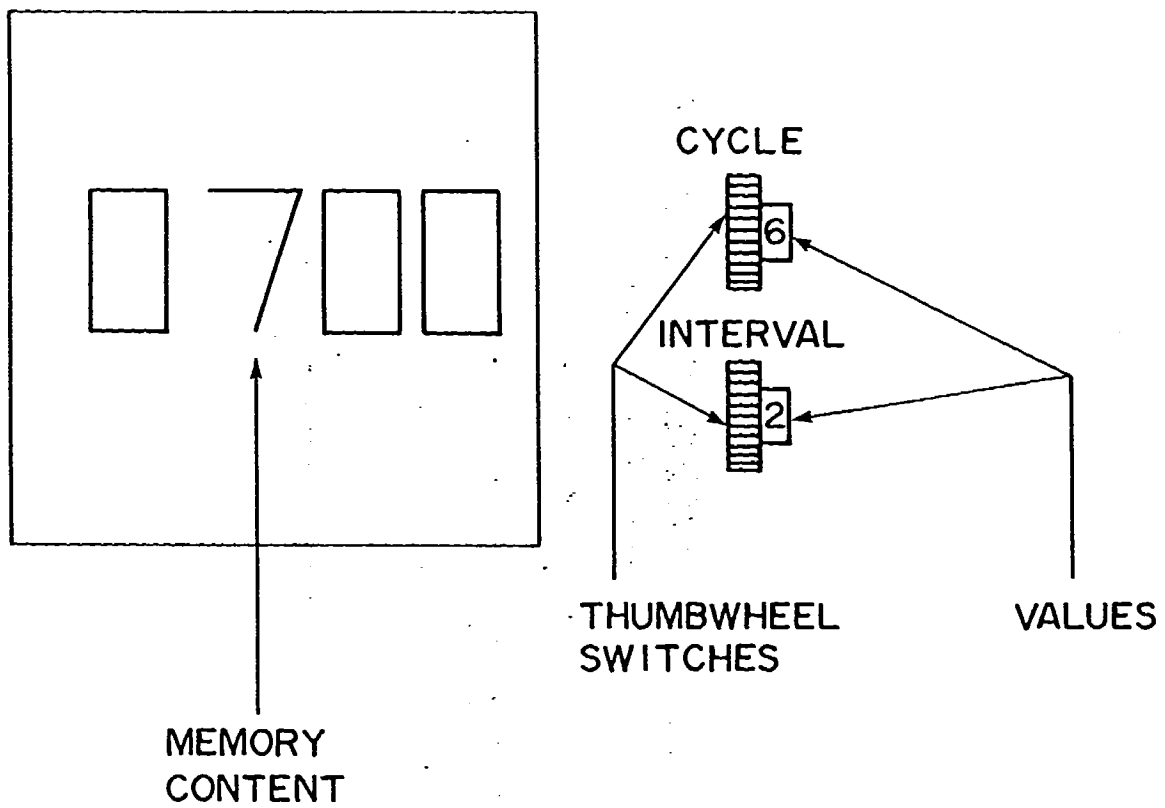


Fig. 3.2 Schematic diagram of ramp controller

Table 3.1 RAMP METERING TIMING CARD

LOCATION _____
 ID NO _____

FE - CYCLE, INT

DIRECTION _____
 RAMP # _____

INTERVAL	CYCLE															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	FE00			FE30												FEF0
1																FEF1
2	FE02			FE32												FEF2
3	FE03															
4																FEF4
5																
6																
7																
8																
9									FE89							
10	A								FE8A							
11	B								FE8B							
12	C															
13	D															
14	E															
15	F	SEC	MIN	HR	WKDY O-6											

FE0F FE1F FE2F FE3F
O = SUNDAY

cycle. Therefore a value of 0050 in FE 30 and a value of 0020 in FE32 will result in 3 seconds RED and 2 seconds GREEN.

Range: 0 to 240.0 in tenths of seconds

FE32 --> FIXED GREEN is the number of seconds of green.
Range: 0 to 240.0 in tenths of seconds

FEF0 --> ID number of the ramp controller at that location.
Range: 0 to 254 entered in whole numbers only.

FEF1 --> WATCHDOG ON TIME is the maximum length of time a detector can remain ON before the watchdog reports that detector as failed.
Range: 0 to 7999 entered in seconds.

FEF2 --> WATCHDOG OFF TIME is the maximum length of time a detector can remain OFF before the watchdog reports that detector as failed.
Range: 0 to 7999 entered in seconds.

FEF4 --> LONG COUNT TIME determines how long the controller will count volumes, violations, occupancy etc. prior to storing them in memory and starting a new count.
Range: 0 to 7999 entered in seconds.

FE89,
FE8A,

FE8B --> These three addresses are used in conjunction and they may be explained as follows.

FE89: The first digit determines the operating pattern:

- 0 ignore the time clock
- 1-6 run cycle chosen
- 7 nonmetering operation
- 8 operate traffic responsive
- 9 ignore the detector watchdog timer

The second digit is always entered as zero. The last two digits select the days of the week the pattern is to operate.

- 00 --> off
- 01 --> Sunday
- 02 --> Monday

- 03 --> Tuesday
- 04 --> Wednesday
- 05 --> Thursday
- 06 --> Friday
- 07 --> Saturday
- 08 --> Monday through Friday
- 09 --> Saturday and Sunday
- 10 --> Tuesday through Thursday
- 11 --> Monday through Friday
- 12 --> Sunday through Saturday.

FE8A: START TIME is a four-digit number representing the time the pattern is to start operating with hours in the first two digits and minutes in the last two digits. Only whole numbers can be entered for minutes.

FE8B: END TIME is a four digit number representing the time the pattern is to stop operating. First two digits are again for hours and the last two digits are for minutes.

FE0F,
FE1F,
FE2F,
FE3F, --> They are respectively the second, minute, hour and weekday (0-6
--> Sunday - Saturday) of the clock used by the ramp controller.

How to Erase and Load Values (Manual Operation)

In order to erase a value in a particular address, the toggle switch is first set to DISPLAY & DETAIL which displays the memory value at the address shown by the thumbwheel switches, i.e., FE Cycle, Interval. If the address is not the one which is desired, it is set using thumbwheel switches. After

the correct address has been set, the E erase key is pressed which erases the value and enters *0000* in that address. Now to load the desired value, the numerical values on the keyboard are used.

It must be remembered that all values entered must be four digits. For example,

240 is entered as 0240

7 is entered as 0007

After the values are loaded, the L load key is pressed firmly. When this is done, care must be taken to see if the LED blinks momentarily, if not it implies that the new values are not loaded. Therefore the load key must be pressed repeatedly until the LED blinks.

Points to remember

- 1) After erasing a value, if no new value is loaded, then the old values are retained.
- 2) If *0000* are to be entered as a new value, then it is not sufficient if the old values are erased. The load key must be pressed so that *0000* displayed by the LED after erasing, are loaded.

Calculation of volume, occupancy and speed

The formula used in the software is shown below, with an explanation of each variable.

Volume calculation

$$VOL(K) = \frac{(SA + PA)}{(E+D)} \times \frac{3600}{ISI}$$

where

VOL = Is the acronym for volume

K = Index of an array which stores calculated volume for all freeway lanes in DATABASS or DATABASN.

SA = Is the total of volume parameters collected by the controller from all or any of the three working secondary detectors.

PA = Is the total of volume parameters collected by the controller from all or any of the four working primary detectors.

E = number of working secondary detectors

D = number of working primary detectors

ISI = Time in sec, for which the data was collected. Also known as the interval time or LONG COUNT TIME.

OCCUPANCY CALCULATION

$$OC = \frac{OPA + OSA}{D+E}$$

where

OSA = Is the total of occupancy parameters collected by the controller from all or any of the three working secondary detectors

OC = occupancy for a particular freeway lane

OPA = Is the total of occupancy parameters collected by the controller from all or any of the four working primary detectors

D = Number of working primary detectors

E = Number of working secondary detectors

Calculation of percent occupancy

$$OCC(K) = \frac{OC}{AMX} * 100$$

where

OCC = Is the acronym for percent occupancy

K = Index of an array which stores calculated percent occupancy for all freeway lanes in DATABASS or DATABASN.

OC = Occupancy for that particular freeway lane

AMX = CT * ISI

where

CT = Occupancy clock frequency in Hertz

ISI = Interval time or LONG COUNT TIME, for which data was collected, in seconds.

Speed calculation

$$V(K) = \frac{DT}{TV} * \frac{3600}{5280} * MF$$

V = Abbreviation for speed

K = Index of an array which stores calculated speed for all freeway lanes in DATABASS or DATABASN.

$$DT = [D1 + RS]$$

where

D1 = Average vehicle length in feet

RS = Range of detectors in feet

$$TV = \left[\frac{OC}{VI} \right] / CT$$

where

OC = occupancy for that particular freeway lane

$VI = \frac{[SA + PA]}{[E + D]}$ variables are the same as explained in volume calculation

CT = occupancy clock frequency in hertz.

3.3 The Ramp Management Strategy (Original)

The decision making process of the Ramp Metering Strategy is shown in Fig. 3.3, Ramp Metering Strategy Flow Diagram, which uses volume, occupancy and speed as parameters.

What is meant by metering

In the ramp management strategy the RED-GREEN cycle used to control traffic is called metering operation. For example, 4sec metering implies 4 seconds RED signal and 2 seconds GREEN signal, and 10 sec metering implies 10 seconds RED signal and 2 seconds GREEN signal. GREEN time is always fixed to 2 seconds in a RED-GREEN cycle. In the revised RMS n sec metering implies n-1 sec RED and 1 sec GREEN.

Address and data to achieve metering

The two most important addresses to achieve the desired RED-GREEN cycle is FE30 and FE32, as previously explained. FE30 is the address for the total cycle parameter and is the RED-GREEN cycle time. FE32 is the address for the GREEN time in the RED-GREEN cycle. Listed below are certain values of RED-GREEN cycle used in the Ramp Management Strategy (original).

SIGNAL	FE30	FE32
Green Signal	0240	0240
Meter 4 secs	0060	0020
Meter 6 secs	0080	0020
Meter 8 secs	0100	0020
Meter 10 secs	0120	0020
Meter 12 secs	0140	0020
Close ramp (red signal)	0240	0000

CHAPTER 4

MODIFIED SOFTWARE

The original Safe-Trans software is not designed for real-time control of the ramp entrances. In order to implement the RMS, the software was modified. In this chapter, the modifications are listed. The new common blocks that store the data and status information are explained. Then the changes made to the main modules - RMPT, COMMT and DATAP, are discussed in detail. Finally, modifications to other routines that figure in other programs, and development of new routines, are discussed briefly. The listing of the complete system is given in the Appendix.

4.1 Common Blocks

COMMON/ASU/ISKIP(2)

COMMON/SMTRC/MDATA(2,10,2,4), IACTR(2,10), IFLAG(2,10,), NFLAG

COMMON/LIEN/MTB(2,10,2,7)

Description:

- | | |
|-----------------|---|
| ISKIP(1) | - The time period for which decision-making won't be implemented by the program when the system is booted up. The unit is second. This prevents spurious data from affecting the RMS. Default value is 600 seconds. |
| ISKIP(2) | - The time period which synchronizes with interval in minutes. The unit is second. |
| MDATA(2,10,2,4) | - The first two digits are used to specify the ramp (southbound from (1,1) to (1,9) northbound from (2,1) to (2,10)). |

The third digit indicates message 1 or message 2. (For every RMS strategy, we need to send two messages to each ramp controller.)

The fourth digit is used as follows:

If the digit is:

1 - done flag, telling COMMT if data needs to be sent (1 means completed formatting, ready to be sent; 0 is no communication).

2 - in combination with the third digit: (1,2) for monitor count use (ref. RMETR), (2,2) for close ramp flag, i.e., if it is 1, ramp is closed, otherwise it isn't.

3 - the address of the message.

4 - the data of the message.

IACR(2,10) - Flag used to indicate if we need to apply the RMS to the specified ramp controller.

IFLAG(2,10) - Contains the last RMS's decision. The purpose of remembering the last RMS's decision is to speed up all the program's functions. If the current decision is the same as the last decision then COMMT will not send message to the ramp controller.

The value in each block represents a different decision:

1 - GREEN

2 - RED

3 - 6 SEC METER

- NFLAG - Indicates whether RMS is in operation or whether the computer system is in data collection mode only.
- MTB(2,10,2,7) - Similar to ITB(2,10,10), used for containing the formatted message for TXMIT to send.

4.2 Modifications to Different Modules

RMPT

Modifications include the following:

1. Addition of the new common blocks to the program.
2. Ask information from operator to get the ramp status values.

IACR(2,10) - active flag for the ramp controllers.

If value is 1, then COMMT will send the data to the ramp controller.

If value is 0, then COMMT will not send the data to the ramp controller.

3. Print out information for validation of input data.

COMMT

Modifications include the following:

1. Before calling PRNT routine, three do loops are added to send the messages out to every ramp controller. It calls MRMT to format the messages and TXMIT to transmit the message to the controller.

2. New Common Block /ASU/, /SMTRC/ and /LIEN/ are attached.

3. New block MTB(2,10,2,7) is used only in the COMMT routine.

It is used to contain the format message for transmission.

DATAP

Modifications include the following:

1. Addition of statements to skip control RMS for the first ten minutes.
2. ISMK routine is called to calculate the volume, occupancy and velocity.
3. New Common Block /ASU/, /SMTRC/ and /LIEN/ are attached.

HISTORY

Modification only includes addition of the new Common Block before DIMENSION statement.

ISMK

ISMK is modified to produce the volume, occupancy and velocity. The reason to do so in this routine is all the collected data will be cleared after the IST file is created. The RMS uses the volume, occupancy and velocity for decision making in RMETR routine. IDD data block contains all the collected data we need to calculate the volume, occupancy and speed. Volume is the maximum number between P(1) and S(1). Occupancy is the maximum number between OP(1) and OS(1). Velocity is calculated from volume and occupancy.

PASCR

It is used to pass the values in the Common block from one program to the other. Before chaining from one program to the other, e.g. from COMMT to DATAP or from DATAP to HISTORY, one external file is used to save the values. The file is "PASC". PASCR routine is used to read all the values back to Common block, so it is called every time a program begins execution.

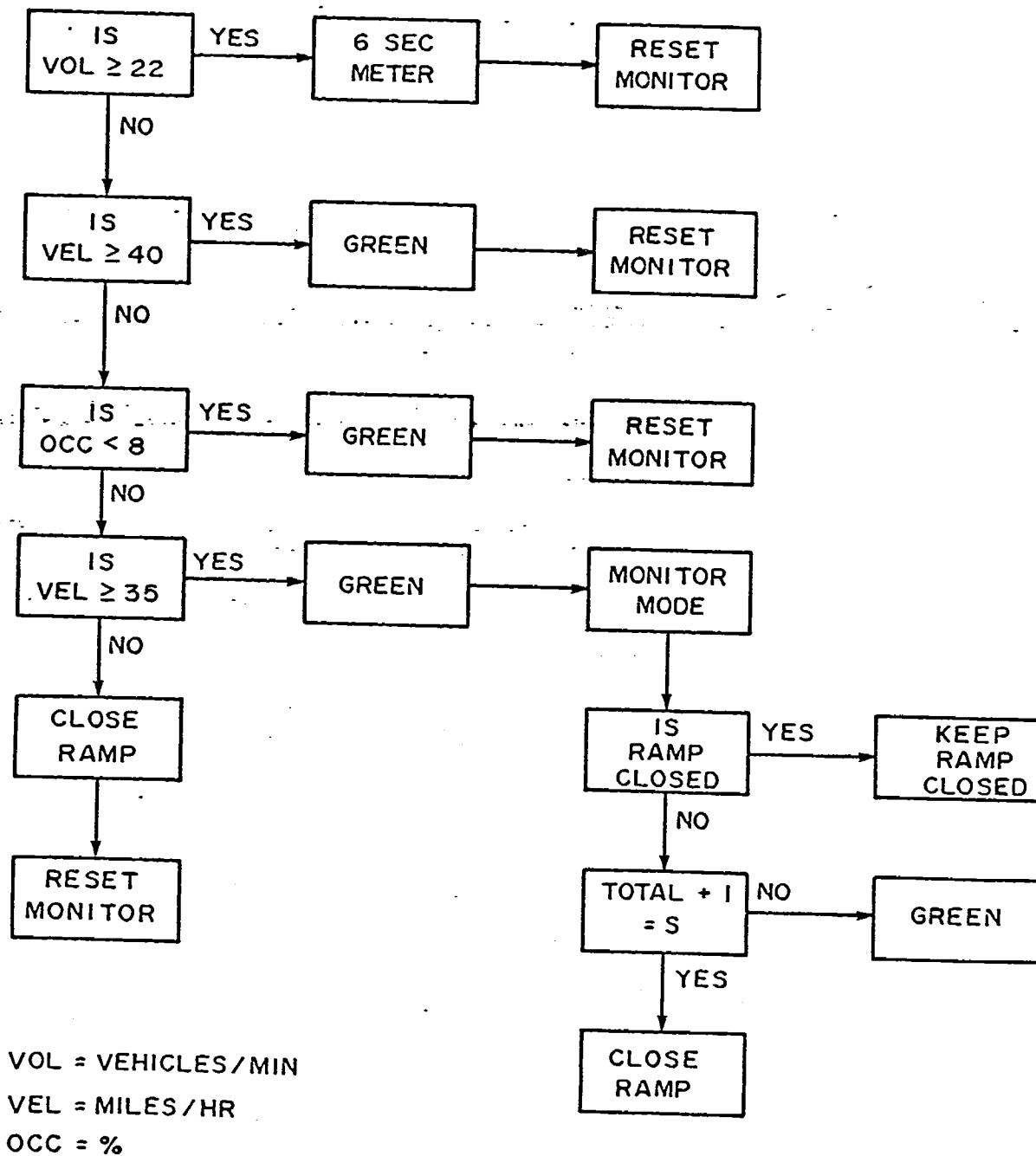


Fig. 4.1 Revised Ramp Management Strategy

PASCH

Similar to PASCR. It writes the common block data to the PASC file.

4.3 New Routines

MRMT

This new routine is used to format the message similar to RMT routine except that it uses fixed IC value (i.e., 16, for write only).

TXMIT

This new routine is used to send the message out without waiting for the response from the ramp controller. It is actually part of SXMITE routine without the receive part. The Common Block used for this routine is MTB(2,10,2,7).

RMETR

RMETR is a revised version of the original RMS developed for this project. RMETR starts metering only if the number of vehicles per hour exceeds 1200 in the outer lane. At any time, depending upon the traffic conditions, the decisions which can be made by this revised strategy are as shown below.

- 1) Rest on green.
- 2) Meter at 6 sec
5 sec red
1 sec green
- 3) Rest on green and monitor.
- 4) Close ramp.

The traffic conditions at which these decisions are made can be easily understood by the flow diagram (Fig. 4.1) provided for

this revised strategy. In order to invoke this subroutine the parameter passed by the calling program are:

- 1) vol - volume
- 2) occ - occupancy
- 3) v - velocity
- 4) I - direction, i.e. northbound or southbound
I = 1 southbound
I = 2 northbound
- 5) J - ramp number

J may assume values from 1 to 10.

To understand the coding for the flow diagram the variable names used in it are explained.

MDATA(i,j,k,l) This is an array with maximum subscript values MDATA (2,10,2,4) where
i = direction (northbound or southbound)
j = ramp number
k,l = for a detailed explanation refer to Chapter 4

IACR(i,j) This is an array with maximum subscript values IACR(2,10) where
i = direction (northbound or southbound)
j = ramp number

Each location in this array may store a value of either 0 or 1, respectively, indicating whether ramp management strategy is to be implemented on that ramp or not.

IFLAG(i,j) This is an array with maximum subscript values IFLAG(2,10) where
i = direction (northbound or southbound)
j = ramp number

Each location in this array may store for a particular ramp a value of either

- 1 for green signal
- 2 for close ramp
- 3 for 6 sec meter

Each decision time, this variable is checked for a ramp to see if the current decision is the same as the previous decision. If it is the same, then the current decision is not sent to that ramp in order to save time.

The original RMS details are omitted here since it was never used in the field tests.

CHAPTER 5

OPERATING PROCEDURES

The step-by-step procedure for operating the RMS system is discussed in the first part of this chapter. The user-system interaction is shown with appropriate user responses. The system flexibility can be seen in the following sections. The modified RAMPTALK program can be used to change most of the parameters that affect the RMS. Finally, the location of all pieces of software are given if changes to the system need to be made at a later date.

5.1 Booting System

1. Make sure you are in the DPOF:RP directory.

Type in "DIR DPOF:RP" to get into RP directory.

2. Type in "Eup@" and CR.
3. Go over to foreground terminal, the printer, and answer the following questions:

Day of the week? (0-6 Sunday - Saturday) _____

Interval in minutes = ? (1-180) answer _____

First interval in minutes = ? (5-180) answer 1

Historical time in minutes = ? (5-180) answer 15

First historical time in minutes = ? (1-4320) answer # minutes
to next quarter-hour.

What ramp will be implemented?

ALL (1)? NONE (2)? SPECIFIED (3)? answer _____

if the answer is 3, it will keep asking

START (1)? MORE (2)? END (3)? answer 1

What ramp number? answer (1010-1029)

START (1)? MORE (2)? END (3)? answer _____

·
·
·

START (1)? MORE (2)? END (3)? answer 3

4. After system is booted up, it will point out the information concerning the first skip interval and skip interval, and the status of the ramp controllers.

5. Wait for 1 minute, then go over to background terminal (console) type in "RAMPTALK" <CR>.

console: "WHAT RAMP NUMBER?"

answer: 1255

console: "ALL"

"CORRECT RAMP? (YES, NO)"

answer: YES

console: "READ, WRITE, START AGAIN, END? (R, W, S, E)"

answer: W

console: "ADDRESS?"

answer: FEF4

console: "DATA?"

answer: answer should be compatible with the answer for "Interval in minutes = ?" (Step 3).

e.g., if answer for "Interval in minutes" is 5, then answer should be $5 * 60 = \underline{0300}$.

if answer for "Interval in minutes" is 2, then answer should be $2 * 60 = \underline{0120}$.

wait for few minutes,

console: "WRITE ALL COMPLETE"

console: "READ, WRITE, START AGAIN, END? (R,W,S,E)"

answer: E

This completes the booting procedure.

5.2 System Changes and Enhancements

Most of the changes have been discussed in this chapter and the preceding chapters. Here the focus is on the new version of RAMPTALK which when used in conjunction with the rest of the system, can give the user control over all the parameters that influence the RMS.

5.3 Directories and Files

All the modified files are in the DPOF:ASU directory and named as follows:

MRMPT - Modified RMPT

MOOMMT - Modified COMMT

MDATAP - Modified DATAP

MHISTORY - Modified HISTORY

LISMK - Modified ISMK

MPASCR - Modified PASCR

MPASOW - Modified PASOW

All the new routines are all in the DPOF:ASU directory:

MRMT, TXMIT, RMETR.

There are also back up disks containing all the modified and new files.

CHAPTER 6

CONCLUDING REMARKS

The current research completes the second phase of the research that began in 1980 with a fixed-time ramp metering system. The original software has been expanded to incorporate strategies, on a real-time basis, to maintain satisfactory flow rates and speed on the freeway. In this chapter, the impact of the ramp management strategy on the freeway traffic is discussed. The discussions are based on the tests carried out by the researchers and ADOT personnel. The report concludes with some thoughts on future enhancements that would make the system more effective.

6.1 Impact on Freeway Traffic

The preliminary tests indicate that ramp metering strategies can be implemented successfully under certain conditions. During peak hour traffic, under test conditions, the system was able to successfully detect the correct volume, occupancy and speed, and implement the strategies in the RMS. In addition to the entry of ramp vehicles according to the speed-volume-occupancy relationships on the freeway, the RMS also provides for ramp closure in the event of a non-recurring incident.

Test 1 (Date: 10/3/85)

This test was carried out on the southbound ramps during morning peak hour traffic. The test began at 7:15 am and continued until 7:45 am at the Glendale and Bethany Home ramps. The revised version of the RMS was used. When the traffic volume reached 1200 vph, the system went into the 6 sec metering mode. This was verified by a manual count of the vehicles on the outer freeway lane. The system did not rest on red as the traffic volume did not exceed 2200 vph and the speed did not fall below 35 mph. However,

since the system was working at 2 min interval (the RMS updated the decisions based on 2 min data), it was felt that the system did not respond adequately to quickly changing conditions on the freeway. These changes were to be incorporated for the next test of the system.

Test 2 (Date: 10/11/85)

This test was carried out during the evening peak hour traffic (3:45 pm to 4:30 pm). The decision-making time was reduced from 2 minutes to 1 minute. As in the previous test, the system went into the 6 sec metering mode when the outer lane volume exceeded 1200 vph. For most of the time, the signal did rest on green at the Bethany Home ramp. At the Glendale ramp the signal finally rested on red. This was caused by an incident on the freeway. A stalled vehicle on the shoulder followed by a Highway Patrol car, slowed the traffic in the outer lane below 35 mph. The signal was made to rest on red for about 10 minutes until the speed and the volume started increasing again.

More tests are planned for the future.

6.2 Future Enhancements

The tests were carried out on a limited number of ramps. Present experience indicates that it takes about 20 seconds to complete one communication sequence with a ramp. Hence a one minute turnaround with the 19 ramps on the system is impossible. The enormous amount of time that is currently consumed is due to several factors listed below.

- (a) The Data General Nova III is an old computer system. The CPU can be easily outperformed even by today's small microcomputers. Since a large number of input/output operations are involved, a considerable amount of time is taken up by the disk operations on obsolete disk

hardware. It is suggested that the computer system be upgraded for faster throughput.

- (b) The original software is written in Fortran V. It is not very efficient and is poorly documented. Furthermore, the modularity is forced by hardware limitations. It is suggested that the language be changed to either FORTRAN 77 or C or a combination of both (where such code mixing is allowed).
- (c) The communication linkup between the main computer and the ramp controllers plays one of the most important roles. At present, the speed (1200 baud) and the reliability (telephone line set-up) are both inadequate. It is suggested that a more reliable and faster system be installed.
- (d) The ramp controller is inflexible in its operation. It cannot be made to rest on red or green in the telemetry mode. Also, apart from the initial boot up, the signal cannot be changed from green to yellow to red, so as to warn the motorist of the changing status. A more versatile controller is required to implement complex ramp metering strategies.

In conclusion, the second phase of the ramp traffic control on the Black Canyon Freeway indicates that an effective system can be installed and made operational with the existing technology.

REFERENCES

- [1] J.B. Blackburn, W.E. Lewis, G.T. Mackulak and R.H. Rucker, "Design of On-Ramp Traffic Control on the Black Canyon Freeway," Final Report, Project No. 190, Arizona Department of Transportation, 198 .
- [2] A. Greist and J. Goosman, Ramp Central Software Manual Arizona, Safetran Traffic Systems, Colorado Springs, CO, May 1980.

PROGRAM LISTINGS

The following pages contain the listing of the FORTRAN V programs used on the Data General NOVA III system. Only the routines that were modified to implement the RMS strategy are shown.

REPORT PREPARED ON 9 28 1985

```
C *****
C **                                RMPT                                **
C **    PRIMARY PROGRAM FOR RAMP CONTROL AND DATA                    **
C **    GATHERING.                                                    **
C *****

COMMON IDAT(2,10,33),KT(3),IRF(2),IRDATA(4),ITS(2,10,10),
+ IFAIL(2,10),IDD(2,10,31,3),INT(5),ICF,IPF,IBUF(2,66)
COMMON/ASU/ ISKIP(2)
COMMON/SMTRC/ MDATA(2,10,2,4),IACTR(2,10),IFLAG(2,10),NFLAG
DIMENSION CMMSG(4)
DOUBLE PRECISION CMMSG

1005    ACCEPT " DAY OF THE WEEK IS ? (0-6 SUNDAY-SATURDAY) ",INT(5)
      IF(INT(5).GT.6)GOTO 1005                                ;WAIT TIL WEEKDAY IS DEFINED
      IF(INT(5).LT.0)GOTO 1005

1006    ACCEPT " INTERVAL IN MINUTES=? (1-180) ",IN
      IF(IN.LE.0)GOTO 1006
      IF(IN.GT.180)GOTO 1006
      INT(2)=IN*60                                           ;SET INTERVAL

1016    ACCEPT " FIRST INTERVAL IN MINUTES=? (1-180) ",IN
      IF(IN.LE.0)GOTO 1016
      IF(IN.GT.180)GOTO 1016
      INT(1)=IN*60                                           ;SET FIRST INTERVAL
      ISKIP(1)=660
      ISKIP(2)=INT(2)

1007    ACCEPT " HISTORICAL TIME IN MINUTES=? (5-180) ",IN
      IF(IN.LE.4)GOTO 1007
      IF(IN.GT.180)GOTO 1007
      INT(4)=IN*60                                           ;SET HISTORY

1017    ACCEPT " FIRST HISTORICAL TIME IN MINUTES=? (1-4320) ",IN
      IF(IN.LE.0)GOTO 1017
      IF(IN.GT.4320)GOTO 1017
      INT(3)=IN*60                                           ;SET FIRST HISTORY
      WRITE(10,29)
29    FORMAT(' WHAT RAMP WILL BE IMPLEMENTED ?')
1019    WRITE(10,30)
30    FORMAT(" ALL(1)? NONE(2)? SPECIFIED(3)? ",Z)
      READ (11,31)IDX
31    FORMAT(I1)
      IF(IDX.EQ.1) GOTO 1020
      IF(IDX.EQ.2) GOTO 1023
      IF(IDX.EQ.3) GOTO 1021
      GO TO 1019
1020    DO 299 I=1,2
      DO 299 J=1,10
299    IACTR(I,J) = 1
      GO TO 1023
1023    NFLAG = 1
      GO TO 1021
1021    WRITE(10,32)
32    FORMAT(" START(1)? MORE(2)? END(3)? ",Z)
      READ(11,31)JDX
      IF(JDX.EQ.1) GOTO 1025
      IF(JDX.EQ.2) GOTO 1025
      IF(JDX.EQ.3) GOTO 1025
1025    WRITE(10,437)
437    FORMAT(" WHAT RAMP NUMBER ? ",Z)
      READ(11,34) NUM
34    9) FORMAT(I4)
      I1 = NUM - 1000
      IF(I1.LT.10) GOTO 1025
      IF(I1.GT.29) GOTO 1025
      I = I1/10
```

```

J = I1-I*10+1
IACTR(I,J) = 1
GO TO 1021
1099 WRITE(10,399)ISKIP(1),ISKIP(2)
399   FORMAT(1X," FIRST SKIP INTERVAL :",I5/
&     1X," SECOND SKIP INTERVAL :",I5)
DO 99 I=1,2
WRITE(10,401)(IACTR(I,J),J=1,10)
401   FORMAT(1X," STATUS OF RAMPS : "/(10I2))
99   CONTINUE
CALL TIME(KT,IER)
CALL SCOMM
CALL SDATM
IC1=0
IC2=0
CALL ERWRT(IC1,IC2)
1010 ITOLD=KT(2)
CALL WAIT(2,2,IER)
CALL TIME(KT,IER)
IF(ITOLD.EQ.KT(2))GOTO 1010
CMSC(1)=3100607
CMSC(2)=" "
CMSC(3)=" "
CMSC(4)=0
CALL ERSTOR(CMSG)
CALL PASCH
CALL FCHAN("COMMT.SU")
STOP
STOP
END
BLOCK DATA
COMMON IDAT(2,10,33),KT(3),IRF(2),IRDATA(4),ITB(2,10,10),
+ IFAIL(2,10),IDD(2,10,31,3),INT(5)
COMMON/ASU/ ISKIP(2)
COMMON/SMTRC/ MDATA(2,10,2,4),IACTR(2,10),IFAIL(2,10),NFLAG
DATA IDAT/660*1/KT/3*0/IRF/0,0/IRDATA/4*0/ITB/200*0/
DATA IFAIL/20*0/IDD/1860*1/INT/60,60,600,600,10/
DATA ISKIP/600,60/
DATA MDATA/201*0/
END

```

REPORT PREPARED ON 9 28 1985

PAGE 3-1-4

```

C *****
C ** COMMUNICATIONS HANDLER **
C *****
COMMON IDAT(2,10,33),KT(3),IRF(2),IRDATA(4),ITB(2,10,10),
+ IFAIL(2,10),IDD(2,10,31,3),INT(5),ICFLG,IPRNTF,IBUF(2,66)
COMMON/ASU/ ISKIP(2)
COMMON/SMTRC/ MDATA(2,10,2,4),IACTR(2,10),IFLAG(2,10),NFLAG
COMMON/LIEN/MTB(2,10,2,7)
DIMENSION CMSG(4),LTIME(3)
DOUBLE PRECISION CMSG
CALL SCONR
CALL PASCR
IDAT(1,10,32)=0
CALL CINIT
C *****
C ** FORMAT MESSAGES **
C *****
IF(NFLAG.EQ.1) GO TO 10
DO 5010 M=1,2
DO 5010 J=1,10
DO 5010 K=1,2
IF(MDATA(1,J,K,1).EQ.0)GOTO 601
I1=-1

```

```

IXB1=13
CALL MRMT(1,J,K)
GO TO 750
601 I1=0
IXB1=0
750 IF(MDATA(2,J,K,1).EQ.0)GOTO 801
I2=-1
IXB2=13
CALL MRMT(2,J,K)
GO TO 950
801 I2=0
IXB2=0
IF((I1.NE.-1).AND.(I2.NE.-1))GOTO 5010
950 CALL TXMIT(I1,I2,J,K,IXB1,IXB2,IER1,IER2)
MDATA(1,J,K,1)=0
MDATA(2,J,K,1)=0
IF(IER1.EQ.1034) MDATA(1,J,K,1)=1
IF(IER2.EQ.1034) MDATA(2,J,K,1)=1
5010 CONTINUE
10 CALL PRNT
DO 1001 J=1,10 ;OUTPUT FROM BACKGROUND
DO 3020 I=1,2 ;DO 10 DROPS
IF(IDAT(1,J,32).NE.0)GOTO 2020 ;DO BOTH LINES
IF(IRDATA(1).LE.0)GOTO 3021 ;NEED DATA?
IF(IRF(1).EQ.1)GOTO 3021 ;NO, RAMP TALK?
IF(IRDATA(2).EQ.1255)GOTO 3000 ;YES, THIS LINE DONE?
J1=IRDATA(2)-1000 ;NO, ALL DROPS?
IF(J1/10.NE.1)GOTO 2900 ;THIS LINE?
J1=J1-J1/10*10 ;YES, DROP #
IF(J1.NE.J-1)GOTO 3021 ;THIS DROP?
IRF(1)=(J)*10 ;YES
GOTO 3010
2900 IRF(1)=1 ;NOT THIS LINE, SET DONE FLAG
GOTO 3020
3000 IF(IRF(1).EQ.0) IRF(1)=5
IRDATA(1)=3
3010 CALL RMT(1,J)
3020 CONTINUE
GOTO 24
3021 ITB(1,J,8)=0 ;NO MESSAGE REQUIRED
ITB(1,J,9)=0 ;CLEAR ALL
ITB(1,J,10)=0 ;FLAGS
GOTO 3020
2020 IF(ITB(1,J,10).EQ.-1)GOTO 3020 ;MESSAGE ALREADY SET UP
I1=(10*I+J-1) ;ID#
ITB(1,J,1)=I1*256+5 ;ADDRESS&BYTES
ICM=128 ;COMMAND
ICSM=I1+5+ICM ;CHECK SUM
ICSMH=(ICSM/256)*256
ICSMH=ICSM-ICSMH

C
C CALL BYTP(ITB(1,J,2),ICM,ICSMH)
C ITB(1,J,3)=ICSMH
C ITB(1,J,8)=66
C ITB(1,J,9)=5
C ITB(1,J,10)=-1
C GOTO 3020
C *****
C ** TRANSMIT & RECEIVE **
C *****
24 I1=ITB(1,J,10)
I2=ITB(2,J,10)
IF((I1.NE.-1).AND.(I2.NE.-1))GOTO 1000 ;NOT REQUIRED THESE DROPS
IXB1=ITB(1,J,9) ;# OF XMIT BYTES
IRB1=ITB(1,J,8) ;# OF RECEIVED BYTES

```

```

IXB2=ITB(2,J,9)
IRB2=ITB(2,J,8)
CALL SXMIT(I1,I2,J,IXB1,IXB2,IRB1,IRB2,IERR1,IERR2)
500   ITB(1,J,10)=1
      ITB(2,J,10)=1
      IF(IRB1.EQ.0) ITB(1,J,10)=0
      IF(IRB2.EQ.0) ITB(2,J,10)=0
      IF(IERR1.NE.1) ITB(1,J,10)=-3
      IF(IERR2.NE.1) ITB(2,J,10)=-3
      IF(IERR1.EQ.1040) ITB(1,J,10)=-2
      IF(IERR2.EQ.1040) ITB(2,J,10)=-2
      DO 2010 I=1,2
      IF(ITB(I,J,10).GT.0) GOTO 100
      IF(ITB(I,J,10).LT.0) GOTO 200
      IF(IRDATA(1).LE.0) GOTO 2010
      IF(IRDATA(2).NE.1255) GOTO 2010
      IRF(I)=IRF(I)-1
      IF(IRF(I).LE.1) IRF(I)=1
2010   CONTINUE
1000   ITOLD=KT(2)                                ;SAVE OLD MINIUTE
      CALL TIME(KT,IER)
      IF(ITOLD.EQ.KT(2)) GO TO 1002
1112   CALL SCOMM
      CALL PASCH
      CALL FCHAN("DATAP.SU")
1002   IF(IRDATA(1).LE.0)GOTO 1001                    ;NEED RAMPTALK?
      IF((IRF(1).NE.1).OR.(IRF(2).NE.1)) GOTO 1001
1003   IRTDATA(1)=0
1004   IRF(1)=0                                       ;YES, RESET LINE 1 FLAG
      IRF(2)=0                                       ;RESET LINE 2 FLAG
      CALL FOPEN(0,"RDATA")                         ;OPEN RAMPTALK FILE
      WRITE BINARY(0)IRDATA                         ;PASS RAMPTALK DATA
      CALL FCLOSE(0)
      IFLG=0
      CALL WRCMN(IFLG,0,1,IER)                       ;TELL RAMPTALK WE ARE DONE
1001   CONTINUE
      CALL WAIT(500,1,IER)                           ;DELAY FOR BACKGROUND
      GOTO 10
C *****
C **      RAMP RESPONSE                                **
C *****
3001   CALL BYTP(IRDATA(4),IBUF(1,6),IBUF(1,5))
      GOTO 1003
C
C
C----- TEST FOR CHECKSUM & MODEL #, THEN PROCESS
C
100   ICSM=0
      DO 210 L=1,64
210   ICSM=ICSM+IBUF(1,L)
      IF(IRF(1)/10.EQ.J) GOTO 3001
      IF(ICSM.NE.(IBUF(1,65)+IBUF(1,66)*256)) GOTO 200
      IF(IBUF(1,2).NE.162) GOTO 200
      IF(IFAIL(1,J).LT.90) GOTO 2056
      CMMSG(1)=3120011                                ;YES, REPORT RESTORED
      CMMSG(2)=" "
      CMMSG(3)=" "
      CMMSG(4)=(10*I+J-1+1000)
      CALL ERSTOR(CMSG)
2056   IFAIL(1,J)=0
      CALL BYTP(IDAT(1,J,1),IBUF(1,4),IBUF(1,3))      ;PACK FOR DATA
      DO 2070 L=56,6,-4
      CALL BYTP(IDAT(1,J,30-(L/2)),IBUF(1,L-2),IBUF(1,L-3))
2070   CALL BYTP(IDAT(1,J,31-(L/2)),IBUF(1,L),IBUF(1,L-1))
      DO 2075 I=56,64,2

```

```

2075 CALL BYTP(IDAT(I,J,(L-1)/2),IBUF(I,L),IBUF(I,L-1))
      IDAT(I,J,33)=0          ;SET GOOD COMM FLAG
      ITS(I,J,10)=0           ;RESET STATUS
      IDAT(I,J,32)=0           ;RESET DATA REQUEST
      GOTO 2010
C *****
C **          COMM ERROR          **
C *****
200  IF(IRF(I)/10.EQ.J)GOTO 2100
      ITS(I,J,10)=0
      IF(IFAIL(I,J).GT.89)GOTO 2092          ;ERSTOR ALREADY
      IFAIL(I,J)=IFAIL(I,J)+1
      IF(IFAIL(I,J).NE.90)GOTO 2091
      CMSC(1)=3120003
      CMSC(2)=" "
      CMSC(3)=" "
      CMSC(4)=(10*I+J-1+1000)
      CALL ERSTOR(CMSC)
2091 IDAT(I,J,32)=1          ;TRY COMM AGAIN
      GOTO 2010
2100 IRDATA(1)=ITS(I,J,10)+1
      GOTO 1004
2092 IDAT(I,J,32)=0          ;FAILED TRY NEXT MINUT
      WRITE(10,999)
999  FORMAT("<BEL>",Z)      ;RING BELL FOR COMM FAIL
      GOTO 2010
      END

```

REPORT PREPARED ON 9 28 1985

```

C *****
C **          MRMT          **
C **          FORMS TRANSMIT MESSAGE          **
C *****

```

```

SUBROUTINE MRMT(I,J,K)
COMMON/SHTRC/ MDATA(2,10,2,4),IACTR(2,10)
COMMON/L IEN/MTB(2,10,2,7)
ID=10*I+J-1
IB=13
LA=MDATA(I,J,K,3)
CALL BYTS(LA,MA)
LD=MDATA(I,J,K,4)
CALL BYTS(LD,MD)
LAN=255-LA
MAN=255-MA
LDN=255-LD
MDN=255-MD
IC=16

```

```

10  CALL BYTP(MTB(I,J,K,1),ID,IB)          ;ID=NUMBER OF BYTES
      MTB(I,J,K,2)=IC*256+LA              ;ADDRESS
      CALL BYTP(MTB(I,J,K,3),MA,LD)
      CALL BYTP(MTB(I,J,K,4),MD,LAN)
      CALL BYTP(MTB(I,J,K,5),MAN,LDN)
      ICS=ID+IB+IC+1020
      MCS=ICS/256*256
      CALL BYTP(MTB(I,J,K,6),MDN,ICS-MCS)
      MTB(I,J,K,7)=MCS
      RETURN
      END

```

REPORT PREPARED ON 9 28 1985

```

C //////////////////////////////////////
C //////////////////////////////////////
C ///          3          -3
C ///
C ///          TRANSMIT
C ///          ROUTINE
C ///
C .....

```

```

SUBROUTINE TXMIT(I1,I2,J,K,IXB1,IXB2,IER1,IER2)
COMMON/SMTRC/ MDATA(2,10,2,4), IACTR(2,10)
COMMON/LIEN/MTB(2,10,2,7)
DIMENSION L1B(15),L2B(15)
----- ITCNT1=30 ----- CHAR TIMER LINE 1
ITCNT2=30 ; " " " 2
IER1=1
IER2=1
I1=I1
I2=I2
TAG=0
IA=1
IB=1
L1=1
L2=1
IXBT1=IXB1
IXBT2=IXB2
CALL CINIT
C- SET UP L1B & L2B
1 L1B(IA+1)=MTB(1,J,K,IB)
CALL BYTS(L1B(IA+1),L1B(IA))
L2B(IA+1)=MTB(2,J,K,IB)
CALL BYTS(L2B(IA+1),L2B(IA))
IA=IA+2
IB=IB+1
IF(IB.LT.8) GO TO 1
IA=1
IB=1
IT1=L1B(1)
IT2=L2B(1)
C- READ STATUS WORD FROM 1600CH
7 CALL INSTAT(ICSTAT)
IF(IXB1.EQ.IXBT1) GOTO 4
C- LOAD NEXT BYTE LINE 1 ;BR IF PREVIOUS BYTE NOT SENT
L1=L1+1
IXBT1=IXB1
IT1=L1B(L1)
4 IF(IXB2.EQ.IXBT2) GOTO 6
C- LOAD NEXT BYTE LINE 2 ;BR IF PREVIOUS BYTE NOT SENT
L2=L2+1
IXBT2=IXB2
IT2=L2B(L2)
C- TRANSMIT LINES 1&2
6 IF(I11.NE.-1) GO TO 3
TAG=TAG+1 ;NO DATA LINE 1
IF(ITEST(ICSTAT,3).NE.0) GOTO 61 ;BR IF READY TO XMIT
ITCNT1=ITCNT1-1 ;DEC TIME OUT
GO TO 3
61 CALL OUTDTS(IT1,4)
IXB1=IXB1-1
ITCNT1=70 ;FIRST LINE
;DEC BYTE CNTR
3 IF(I12.NE.-1) GO TO 5 ;RESET TIME OUT
IF(ITEST(ICSTAT,5).NE.0) GOTO 31 ;NO DATA LINE 2
ITCNT2=ITCNT2-1 ;BR IF READY TO XMIT
GO TO 5 ;DEC TIME OUT
31 CALL OUTDTS(IT2,16)
IXB2=IXB2-1
ITCNT2=70 ;XMIT SECOND LINE
;DEC BYTE COUNTER
5 IF((IXB1.LE.0).OR.(ITCNT1.LE.0)) I11=0 ;RESET TIME OUT
IF((IXB2.LE.0).OR.(ITCNT2.LE.0)) I12=0 ;LINE 1 FINISHED
IF(ITCNT1.EQ.0) IER1=1034 ;LINE 2 FINISHED
IF(ITCNT2.EQ.0) IER2=1034 ;TIME OUT CODE
IF((I11.EQ.0).AND.(I12.EQ.0)) GOTO 900
GO TO 7
900 CALL WAIT(250,1,IER)
;WAIT FOR MDMCM

```

RETURN
END

REPORT PREPARED ON 9 28 1985

```
C *****
C **                                DATAP                                **
C **                                DATA PROCESSOR                        **
C **                                CONVERTS DATA FROM RAMPS INTO VOLUMES AND **
C **                                OCCUPANCIES.  BUILDS ISFILES.  BUILDS HIS0 **
C **                                FILE.  CALLS HISTORY AND SYNC.          **
C *****

COMMON IDAT(2,10,33),KT(3),IRF(2),IRDATA(4),ITB(2,10,10),
+ IFAIL(2,10),IOD(2,10,31,3),INT(5)
COMMON/ASU/ ISKIP(2)
COMMON/SMTRC/ MDATA(2,10,2,4),IACR(2,10),IFLAG(2,10),NFLAG
DOUBLE PRECISION CMSC
DIMENSION ITIME(3),JTIME(3),KTT(5),KD(3),CMSC(4),CM(13)
CM(1)="P1"
CM(2)="P2"
CM(3)="P3"
CM(4)="P4"
CM(5)="S1"
CM(6)="S2"
CM(7)="S3"
CM(8)=" Q"
CM(9)="C1"
CM(10)="C2"
CM(11)=" R"
CM(12)="F1"
CM(13)="F2"
CALL PASCR
CALL SDATR
CALL ERREC(ILCNT1,ILCNT2)
IFCNT2=0
IFCNT1=0
DO 1 J=1,10
IFCNT1=IDAT(1,J,33)+IFCNT1
IFCNT2=IDAT(2,J,33)+IFCNT2
1  CONTINUE
IF(IFCNT1.LT.5) GO TO 2
ILCNT1=ILCNT1+1
;LINE TIMER
GOTO 3
2  IF(ILCNT1.GE.4) GOTO 15
ILCNT1=0
3  IF(IFCNT2.LT.5) GO TO 5
ILCNT2=ILCNT2+1
GOTO 4
5  IF(ILCNT2.GE.4) GOTO 11
ILCNT2=0
4  IF(ILCNT1.EQ.4) GOTO 6
9  IF(ILCNT2.EQ.4) GOTO 7
GOTO 9
6  CMSC(1)=3061203
CMSC(2)="LINE # "
CMSC(3)="ONE"
CMSC(4)=0
CALL ERSTOR(CMSC)
GOTO 8
7  CMSC(1)=3061203
CMSC(2)="LINE # "
CMSC(3)="TWO"
CMSC(4)=0
CALL ERSTOR(CMSC)
GOTO 9
15 CMSC(1)=3061211
CMSC(2)="LINE # "
CMSC(3)="ONE"
```

```

15 MSG(1)=3061211
MSG(2)="LINE # "
MSG(3)="ONE"
MSG(4)=0
CALL ERSTOR(MSG)
ILCNT1=0
GOTO 3

11 MSG(1)=3061211
MSG(2)="LINE # "
MSG(3)="TWO"
MSG(4)=0
CALL ERSTOR(MSG)
ILCNT2=0
GOTO 4

9 CONTINUE
DO 100 I=1,2
DO 99 J=1,10
IF(IDAT(I,J,33).NE.0)GOTO 96
IDD(I,J,28,3)=0
IDAT(I,J,33)=1
DO 90 K=2,26,+2
CALL BCDB(IDAT(I,J,K))
IVOL=IDAT(I,J,K)-IDD(I,J,K,2)
IF(IVOL.LT.0)IVOL=IVOL+10000
IDD(I,J,K,3)=IDD(I,J,K,3)+IVOL
IDD(I,J,K,1)=IDD(I,J,K,1)+IVOL
IDD(I,J,K,2)=IDAT(I,J,K)
IDD=IDD(I,J,K+1,2)
ION=IDAT(I,J,K+1)
IF((IDD.GT.0).AND.(ION.LT.0))GOTO 80
IOCC=ION-IDD
IF(IOCC.LT.0)IOCC=0
89 IDD(I,J,K+1,3)=IDD(I,J,K+1,3)+IOCC
IDD(I,J,K+1,1)=IDD(I,J,K+1,1)+IOCC
90 IDD(I,J,K+1,2)=IDAT(I,J,K+1)
CONTINUE
IF(IDD(I,J,29,3).EQ.0)GOTO 96
IDD(I,J,29,3)=0
DO 95 K=2,27
IDD(I,J,K,3)=0
95 IDD(I,J,K,1)=0
IDD(I,J,29,2)=0
96 IDAT(I,J,32)=1
98 IF(IDAT(I,J,29)-IDD(I,J,29,2)) 300,99,300
300 IDEO=IDD(I,J,29,2)
IDEN=IDAT(I,J,29)
ID=NOT.(IDEO.AND.IDEN)
IDER=(IDEO.AND.ID)
IDEF=(IDEN.AND.ID)
MSK=1
DO 310 K=1,13
IF((IDER.AND.MSK).NE.0) GOTO 320
IF((IDEF.AND.MSK).NE.0) GOTO 330
310 MSK=MSK*2
GO TO 350
320 MSG(1)=3000111
GOTO 340
330 MSG(1)=3000103
340 MSG(3)=CM(K)
MSG(2)=" "
MSG(4)=10*I+J-1+1000
CALL ERSTOR(MSG)
GOTO 310
350 IDD(I,J,29,2)=IDEN
99 CONTINUE
100 CONTINUE

;NO COMM THIS TIME
;FLAG FOR ISMK
;RESET COMM FLAG
;EACH DETECTOR
;CONVERT BCD TO BINARY
;DELTA VOL
;CORRECT OVERFLOW
;UPDATES INTERVAL BUFFER
;UPDATES HISTORY BUFFER
;FOR NEXT TIME
;OLD OCC
;NEW OCC
;OVERFLOW HAPPENED
;DELTA OCC
;UPDATES INTERVAL BUFFER
;UPDATES HISTORY BUFFER
;FOR NEXT TIME
;FIRST TIME THIS DROP?
;YES, RESET FLAG
;ZERO INTERVAL AND HISTORY
;CLEAR DETECTOR ERRORS
;REQUEST NEW DATA
;OLD DETECTOR ERRORS
;NEW ERRORS
;FAILED DET
;RESTORED
;JUST FAILED
;SHIFT TO LEFT
;DET RESTORED
;DET FAILED
;GET DET NAME
;RAMP NO.
;SET ERROR
;SET NEW ERRORS

```



```

100 CONTINUE
102 CALL TIME(KT, IER)
   ISKIP(1)=ISKIP(1)-60
   INT(1)=INT(1)-60
   IF(INT(1).GT.0)GOTO 150
   INT(1)=INT(2)
   IDD(1,1,1,3)=INT(1)
   CALL DATE(KD, IER)
200   CALL WAIT(10,1,IER)
   CALL OPEN(1,"DATABSM",1,IER)
   IF(IER.NE.1)GOTO 200
205   CALL ISMK
105   CALL WAIT(10,1,IER)
   CALL DFILW("ISFILEM",IER)
   IF(IER.EQ.51)GOTO 105
119   CALL WAIT(10,1,IER)
220   CALL RENAME("ISFILEM","ISFILEM",IER)
   IF(IER.EQ.51)GOTO 119
225   CALL RENAME("IST","ISFILEM",IER)
230   CALL WAIT(10,1,IER)
   CALL OPEN(1,"DATABASE",1,IER)
   IF(IER.NE.1)GOTO 230
235   CALL ISMK
   IF(ISKIP(1).GT.0)GOTO 110
   ISKIP(1)=ISKIP(2)
110   CALL WAIT(10,1,IER)
   CALL DFILW("ISFILES1",IER)
   IF(IER.EQ.51)GOTO 110
149   CALL WAIT(10,1,IER)
   CALL RENAME("ISFILES","ISFILES1",IER)
   IF(IER.EQ.51)GO TO 149
255   CALL RENAME("IST","ISFILES",IER)
150   INT(3)=INT(3)-60
   IF(INT(3).GT.0)GOTO 10
   INT(3)=INT(4)
   CALL HISMK
   OPEN 1,"HSMK0"
171   CALL APPEND(2,"HIS0",0,IER)
   IF(IER.NE.1) CALL WAIT(10,1,IER)
   IF(IER.NE.1) GO TO 171
170   READ BINARY(1,END=160)I
   WRITE BINARY(2)I
   GOTO 170
160   CLOSE 2
   CLOSE 1
   CALL DFILW("HSMK0",IER)
C *****
C **                               CLKUP                               **
C **   READS BACKUP CLOCK AND COMPARES WITH SYSTEM                     **
C **   CLOCK. IF SYSTEM CLOCK IS BEHIND IT IS SET                      **
C **   EQUAL TO THE BACKUP. IF SYSTEM IS AHEAD THE                     **
C **   DATE IS ADVANCED ONE DAY AND THE CLOCK IS THEN                  **
C **   SET EQUAL TO THE BACKUP.                                         **
C *****
10   CALL RCLK(IHOUR,IMINSEC)
   CALL WAIT(10,1,IER)
   CALL RCLK(itime(1),itime(3))
   IF (IMINSEC.NE.itime(3))GOTO 10
   IF(IHOUR.NE.itime(1))GOTO 10
   CALL BYTS(itime(3),itime(2))
   CALL BYTS(itime(1),IDUM)
   CALL BCDB(itime(3))
   CALL BCDB(itime(2))
   CALL BCDB(itime(1))
   CALL TIME(JTIME,IER)
   IF(JTIME(3).LT.10)GOTO 155

```

;INTERVAL TIMER
 ;INTERVAL DONE?
 ;YES, RESET TIMER

 ;DELETE OLD

 ;CURRENT FILE =LAST
 ;NEW CURRENT FILE

 ;HISTORY TIMER
 ;HISTORY DONE?
 ;YES, RESET TIMER

 ;READ THE BACKUP CLOCK
 ;WAIT 10 MSEC
 ;READ THE CLK AGAIN
 ;STILL THE SAME?

 ;SEPARATE MIN/SEC
 ;SEPARATE HOUR
 ;CONVERT BCD SEC TO BINARY
 ;CONVERT BCD MIN TO BINARY
 ;CONVERT BCD HOUR TO BINARY
 ;GET CURRENT SYSTEM TIME
 ;TOO CLOSE TO MINUTE CHANGE

```

CALL TIME(JTIME,IER)
IF(JTIME(3).LT.10)GOTO 155
IF(JTIME(3).GT.50)GOTO 155
IF(JTIME(1).GT.ITIME(1))GOTO 2040
IF(JTIME(1).NE.ITIME(1))GOTO 2000
IF(JTIME(2).NE.ITIME(2))GOTO 2000
IF(ITIME(3)-JTIME(3).GT.2)GOTO 2000
GOTO 1002

```

```

;GET CURRENT SYSTEM TIME
;TOO CLOSE TO MINUTE CHANGE
;
;SYSTEM HOURS>BACKUP HOURS
;SYSTEM HOURS NE BACKUP HOURS
;SYSTEM MIN NE BACKUP MIN
;SYSTEM SEC NE BACKUP SEC

```

2000 CMSG(1)=3100403

```

CMSG(2)="
CMSG(3)="
CMSG(4)="

```

2011 CALL ERSTOR(CMSG)

```

CALL RCLK(IHOUR,IMINSEC)
CALL WAIT(10,1,IER)
CALL RCLK(ITIME(1),ITIME(3))
IF(IMINSEC.NE.ITIME(3))GOTO 2011
IF(IHOUR.NE.ITIME(1))GOTO 2011

```

```

;READ THE BACKUP CLOCK
;WAIT 10 MSEC
;READ THE CLK AGAIN
;STILL THE SAME?
;YES

```

```

CALL BYTS(ITIME(3),ITIME(2))
CALL BYTS(ITIME(1),IDUM)
CALL BCD8(ITIME(3))
CALL BCD8(ITIME(2))
CALL BCD8(ITIME(1))
CALL STIME(ITIME,IER)

```

```

;SEPARATE MIN/SEC
;SEPARATE HOUR
;CONVERT BCD SEC TO BINAR
;CONVERT BCD MIN TO BINARY
;CONVERT BCD HOUR TO BINARY

```

KT(1)=ITIME(1)

KT(2)=ITIME(2)

KT(3)=ITIME(3)

CMSG(1)=3100411

CALL ERSTOR(CMSG)

2097 CALL WAIT(1,1,IER)

CALL OPEN(5,"#TT01",1,IER)

IF(IER.NE.1)GOTO 2097

WRITE(5,2100)

WRITE(5,2101)

WRITE(5,2102)ITIME(1),ITIME(2),ITIME(3),KD(1),KD(2),KD(3)

CLOSE 5

2100 FORMAT(1X,"POWER FAIL HAS OCCURED")

2101 FORMAT(1X,"SYSTEM TIME IS")

2102 FORMAT(1X,I2,":",I2,":",I2,4X,I2,"/",I2,"/",I4)

GOTO 1001

2040 ITIME(1)=23

ITIME(2)=59

ITIME(3)=59

1002 CALL STIME(ITIME,IER)

;SET TIME=23 59 59

KT(1)=ITIME(1)

KT(2)=ITIME(2)

KT(3)=ITIME(3)

C *****

C ** HISTORY

C ** UPDATES HISTORY FILES WHENEVER THE CURRENT

C ** DATE IS NEWER THAN THE DATE OF HIS0

C ** FILE. HIS31 IS DELETED AND A NEW HIS0

C ** CREATED.

C *****

155 CALL DATE(KD,IER)

OPEN 0,"HIS0"

READ (0,END=71)IMON,IDAY,IYR

CLOSE 0

IF(KD(3)-IYR)1001,65,120

65 IF(KD(1)-IMON)1001,75,120

75 IF(KD(2)-IDAY)1001,1001,120

126 CALL PASCH

;DAY HAS CHANGED

CALL SDATM

;CALL HISTORY

CALL FCHAN("HISTORY.SU")

71 WRITE(0)KD(1),KD(2),KD(3)

CLOSE 0

CALL SDATW

CALL HISTORY

CALL FCHAN("HISTORY.SU")

71 WRITE(0)KD(1),KD(2),KD(3)

CLOSE 0

1001 CALL PASCH

CALL SDATW

CALL ERWRT(ILCNT1,ILCNT2)

CALL FCHAN("COMMT.SU")

80 IOCC=(32767-100)+(32767+100)+2

GOTO 89

END

REPORT PREPARED ON 9 28 1985

SUBROUTINE RMETR(VOL,OCC,U,I,J)

COMMON/SMTRC/MDATA(2,10,2,4),IACR(2,10),IFLAG(2,10)

WRITE(10,666)IACR(I,J)

666 FORMAT(4X,I2,Z)

IF((I.EQ.1).AND.(J.EQ.9)) GO TO 1000

IF(VOL.NE.0.0) GO TO 1000

MDATA(I,J,1,1) = 0

MDATA(I,J,2,1) = 0

WRITE(10,2345)

2345 FORMAT(' NO COMMUN.')

GO TO 777

1000 MDATA(I,J,1,1) = 1

MDATA(I,J,2,1) = 1

IF(IACR(I,J).EQ.0) GO TO 702

MDATA(I,J,1,3) = -464

MDATA(I,J,2,3) = -462

GO TO 703

702 MDATA(I,J,1,3) = -349

MDATA(I,J,2,3)=-333

703 VOL = VOL/54.6

IF(VOL.LT.22.0)GOTO 19

WRITE(10,6)

6 FORMAT(' 6SEC METER')

IF(IFLAG(I,J).EQ.3) GOTO 888

IFLAG(I,J)=3

MDATA(I,J,1,4) = 96

MDATA(I,J,2,4) = 32

MDATA(I,J,1,2) = 0

MDATA(I,J,2,2) = 0

GO TO 777

19 IF (V.LT.40.0) GOTO 21

25 WRITE(10,3)

3 FORMAT(' GREEN')

IF(IFLAG(I,J).EQ.1) GOTO 888

IFLAG(I,J)=1

MDATA(I,J,1,4) = 0

MDATA(I,J,2,4) = 576

MDATA(I,J,1,2) = 0

MDATA(I,J,2,2) = 0

GO TO 777

21 IF(OCC.GT.8.0) GOTO 37

GO TO 25

37 IF(V.GT.35.0) GOTO 51

15 MDATA(I,J,2,2) = 1

WRITE(10,4)

4 FORMAT(' CLOSE RAMP')

IF(IFLAG(I,J).EQ.2) GOTO 888

IFLAG(I,J)=2

MDATA(I,J,1,4) = 288

MDATA(I,J,2,4) = 0

MDATA(I,J,1,2) = 0

GO TO 777

51 ICLOSE = MDATA(I,J,2,2)

IF(ICLOSE.EQ.1) GO TO 15

MDATA(I,J,1,2) = MDATA(I,J,1,2) + 1

```

      IF(ICLOSE.EQ.1) GO TO 15
      MDATA(I,J,1,2) = MDATA(I,J,1,2) + 1
      ICOUNT = MDATA(I,J,1,2)
      IF(ICOUNT.GE.5) GO TO 15
      GO TO 25
888      MDATA(I,J,1,1)=0
      MDATA(I,J,2,1)=0
777      RETURN
      EH
      REPORT PREPARED ON 9 28 1985
C *****
C ** HISTORY **
C ** ONCE EACH DAY THE HISTORY FILES ARE **
C ** MOVED AND THE CLOCK SYNC CALLED **
C *****
      COMMON IDAT(2,10,33),KT(3),IRF(2),IRDATA(4),ITB(2,10,10),
+ IFATL(2,10),IDG(2,10,31,3),INT(5)
      COMMON/ASU/ ISKIP(2)
      COMMON/ENTRC/ MDATA(2,10,2,4),IACTR(2,10),IFLAG(2,10),NFLAG
      DIMENSION KD(3),ICLKDTA(11)
      CALL PASCR
      CALL SCOMR
      CALL SDATR
      CALL DATE(KD,IER)
      CALL DIR("ARIZ",IER)
      IF(IER.EQ.1)GOTO 10
      WRITE(10,11)
11      FORMAT(" CAN NOT FIND DPO:ARIZ. MADE HIS1 IN RP.")
10      CALL DFILW("HIS31",IER)
      CALL RENAME("HIS30","HIS31",IER)
      CALL RENAME("HIS29","HIS30",IER)
      CALL RENAME("HIS28","HIS29",IER)
      CALL RENAME("HIS27","HIS28",IER)
      CALL RENAME("HIS26","HIS27",IER)
      CALL RENAME("HIS25","HIS26",IER)
      CALL RENAME("HIS24","HIS25",IER)
      CALL RENAME("HIS23","HIS24",IER)
      CALL RENAME("HIS22","HIS23",IER)
      CALL RENAME("HIS21","HIS22",IER)
      CALL RENAME("HIS20","HIS21",IER)
      CALL RENAME("HIS19","HIS20",IER)
      CALL RENAME("HIS18","HIS19",IER)
      CALL RENAME("HIS17","HIS18",IER)
      CALL RENAME("HIS16","HIS17",IER)
      CALL RENAME("HIS15","HIS16",IER)
      CALL RENAME("HIS14","HIS15",IER)
      CALL RENAME("HIS13","HIS14",IER)
      CALL RENAME("HIS12","HIS13",IER)
      CALL RENAME("HIS11","HIS12",IER)
      CALL RENAME("HIS10","HIS11",IER)
      CALL RENAME("HIS9","HIS10",IER)
      CALL RENAME("HIS8","HIS9",IER)
      CALL RENAME("HIS7","HIS8",IER)
      CALL RENAME("HIS6","HIS7",IER)
      CALL RENAME("HIS5","HIS6",IER)
      CALL RENAME("HIS4","HIS5",IER)
      CALL RENAME("HIS3","HIS4",IER)
      CALL RENAME("HIS2","HIS3",IER)
      CALL RENAME("HIS1","HIS2",IER)
      OPEN 0,"HIS1"
      OPEN 1,"RP:HIS0"
60      READ BINARY(1,END=70)I 3
      WRITE BINARY(0)I
      GOTO 60
70      CLOSE 0
      CLOSE 1
      CALL DIR("RP",IER)

```

```

CLOSE 1
CALL DIRC("RP",IER)
CALL DFILN("HIS0",IER)
OPEN 0,"HIS0"
WRITE(0)KD(1),KD(2),KD(3)
CLOSE 0
C *****
C **
C **      ONCE EACH DAY AFTER HISTORY IS UPDATED,
C **      SECONDS,MINUTES,HOURS,DAYS AND LONG
C **      COUNTER ARE SYNCRONIZED AT ALL CONTROLLERS.
C *****
C *****
INT(5)=INT(5)+1
IF(INT(5).GT.6)INT(5)=0
CALL TIME(KT,IER)
ICLKDTA(2)=-497
CALL BBOD(KT(3))
ICLKDTA(3)=KT(3)+14
ICLKDTA(4)=-76
ICLKDTA(5)=0
ICLKDTA(6)=-481
CALL BBOD(KT(2))
ICLKDTA(7)=KT(2)
ICLKDTA(8)=-465
CALL BBOD(KT(1))
ICLKDTA(9)=KT(1)
ICLKDTA(10)=-449
ICLKDTA(11)=INT(5)
C *****
C **      CKSYNC
C **      SETS UP FIVE CLOCK SYNC MESSAGES ONE AT A TIME
C **      ON EACH COMM LINE
C *****
ICNT=0
IC=1
IM=5
DO 21 I=1,2
DO 22 J1=IC,IM
ITB(I,J1,1)=-243
ICMD=4096
LA=ICLKDTA(12-2*(6-J1))
CALL BYTS(LA,MA)
ITB(I,J1,2)=ICMD+LA
LD=ICLKDTA(13-2*(6-J1))
CALL BYTS(LD,MD)
CALL BYTP(ITB(I,J1,3),MA,LD)
LAN=255-LA
CALL BYTP(ITB(I,J1,4),MD,LAN)
LDN=255-LD
MAN=255-MA
CALL BYTP(ITB(I,J1,5),MAN,LDN)
MDN=255-MD
CALL BYTP(ITB(I,J1,6),MDN,24)
ITB(I,J1,8)=0
ITB(I,J1,9)=13
ITB(I,J1,7)=1280
ITB(I,J1,10)=-1
CONTINUE
ICNT=ICNT+1
IF(ICNT.GE.2) GOTO 31
IC=6
IM=10
GO TO 30
31      ICNT=0
IC=1

```

REPORT PREPARED ON 9 28 1985

REPORT PREPARED ON 9 28 1985

REPORT PREPARED ON 9 28 1985

```

SUBROUTINE ISMK
COMMON IDAT(2,10,33),KT(3),IRF(2),IRDATA(4),ITB(2,10,10),
+ IFAIL(2,10),IDD(2,10,31,3),INT(5)
COMMON/ASU/ ISKIP(2)
COMMON/SMTRC/ MDATA(2,10,2,4),IACTR(2,10),IFLAG(2,10),NFLAG
DIMENSION ITIM(3),IDATE(3),IDP(4),ISP(3),IP(4),IS(3),KD(3)

CALL DATE(KD,IER)
READ FREE(1)INUM
FORMAT(I6)
OPEN 2, "IST"
N=1
DO 201 N=1,INUM
READ(1,7)J1,J2,J3,J4
FORMAT(S8,2I4,I7)
J3=J4-1000
I=J3/10
J=J3-10*I+1
J2=ITEST(IDAT(I,J,31),3)
WRITE(2)KD(1),KD(2),KD(3),KT(1),KT(2),KT(3),IDD(1,1,1,3),J2,J4
K=IDAT(I,J,29)
DO 200 L=1,13
IF(IDD(I,J,28,3).NE.0) GOTO 205
L1=ITEST(K,13-L)

```

```

205 IF(IDD(I,J,28,3).NE.0) GOTO 205
    L1=ITEST(K,13-L)
    IF(L1.EQ.0) GOTO 200
    IDD(I,J,2*L,3)=-1
    IDD(I,J,2*L+1,3)=-1
200 CONTINUE
    IDD(I,J,28,3)=1
    DO 210 M1=2,26,+2
210 WRITE(2)IDD(I,J,M1,3)
    DO 220 M1=3,27,+2
220 WRITE(2) IDD(I,J,M1,3)
    CALL BCDB(IDAT(I,J,28))
    WRITE(2) IDAT(I,J,28)
    IF(ISKIP(1).GT.0) GOTO 1000
    IF(NFLAG.EQ.1) GOTO 1000
    IF(N.NE.1) GOTO 50
    WRITE(10,6)
    WRITE(10,8)
6    FORMAT(' RAMP',6X,'DATE',8X,'TIME',6X,
&    'VOL',4X,'OCC SPEED',2X,'STATUS RMS')
8    FORMAT(' NO',29X,'UPH',4X,'(%)',4X,'MPH')
50    CALL DATE(IDATE,IER)
    ITIM(1)=KT(1)
    ITIM(2)=KT(2)
    ITIM(3)=KT(3)
    ISI=IDD(1,1,1,3)
    IP(1)=IDD(1,J,26,3)
    IS(1)=IDD(1,J,18,3)
    IOP(1)=IDD(1,J,27,3)
    ISP(1)=IDD(1,J,19,3)
    PA=IP(1)
    SA=IS(1)
    U1=AMAX1(PA,SA,0.0)
    IF(U1.EQ.0.0) GOTO 113
    VOL=(U1/ISI)*3600*0.91
    GO TO 115
113 VOL=0.
115 OPA=IOP(1)
    OSA=ISP(1)
    OC=AMAX1(OPA,OSA,0.0)
    IF(OC.EQ.0.0) GOTO 119
    GO TO 121
119 OC=0.
121 AMX=30.*ISI
    OCC=(OC/AMX)*100
    IF(U1.EQ.0.0) U=0.
    IF(U1.EQ.0.0) GO TO 99
    TU=(OC/U1)/30.
    DT=22.
    U=(DT/TU)*(3600./5280.)
100 CONTINUE
99 WRITE(10,2) J4,IDATE,ITIM,VOL,OCC,U
2    FORMAT(15,214,15,15,213,F8.0,F7.0,F7.0,2)
    CALL RMETR(VOL,OCC,U,1,J)
    IF(N.NE.INUM) GOTO 1000
    WRITE(10,4)
4    FORMAT(5X,' ')
1000 DO 202 L=2,27
202 IDD(1,J,L,3)=0
    N=N+1
201 CONTINUE
    IDD(1,1,1,3)=INT(1)
    CLOSE 2
    CLOSE 1
    RETURN

```

201 CONTINUE
1.4 ADD(1,1,1,3)=INT(1)
CLOSE 2
CLOSE 1
RETURN
END